# PGE: Robust Product Graph Embedding Learning for Error Detection

Kewei Cheng
viviancheng@cs.ucla.edu
University of California, Los Angeles

Xian Li
xianlee@amazon.com
Amazon.com

Yifan Ethan Xu
xuyifa@amazon.com
Amazon.com

Xin Luna Dong
lunadong@gmail.com
Facebook.com

Yizhou Sun
yzsun@cs.ucla.edu
University of California, Los Angeles

## ABSTRACT

Although product graphs (PGs) have gained increasing attentions in recent years for their successful applications in product search and recommendations, the extensive power of PGs can be limited by the inevitable involvement of various kinds of errors. Thus, it is critical to validate the correctness of triples in PGs to improve their reliability. Knowledge graph (KG) embedding methods have strong error detection abilities. Yet, existing KG embedding methods may not be directly applicable to a PG due to its distinct characteristics: (1) PG contains rich textual signals, which necessitates a joint exploration of both text information and graph structure; (2) PG contains a large number of attribute triples, in which attribute values are represented by free texts. Since free texts are too flexible to define entities in KGs, traditional way to map entities to their embeddings using ids is no longer appropriate for attribute value representation; (3) Noisy triples in a PG mislead the embedding learning and significantly hurt the performance of error detection. To address the aforementioned challenges, we propose an end-to-end noise-tolerant embedding learning framework, PGE, to jointly leverage both text information and graph structure in PG to learn embeddings for error detection. Experimental results on real-world product graph demonstrate the effectiveness of the proposed framework comparing with the state-of-the-art approaches.

**PVLDB Reference Format:**
Kewei Cheng, Xian Li, Yifan Ethan Xu, Xin Luna Dong, and Yizhou Sun. PGE: Robust Product Graph Embedding Learning for Error Detection. PVLDB, 15(6): 1288 - 1296, 2022.
doi:10.14778/3514061.3514074

## 1 INTRODUCTION

With the rapid growth of the internet, e-commerce websites such as Amazon, eBay, and Walmart provide important channels to facilitate online shopping and business transactions. As an effective way to organize product-related information, product knowledge graphs (PGs) [14] have attracted increasing attentions in recent years by empowering many real-world applications, such as product search and recommendations [2].

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 15, No. 6 ISSN 2150-8097.
doi:10.14778/3514061.3514074

| Title | Category | Flavor | Ingredient |
|---|---|---|---|
| Brand A Tortilla Chips Spicy Queso, 6 - 2 oz bags † | chips-and -crisps | Spicy Queso | Ground Corn, Chipotle Pepper Powder, Paprika Extract, Spices |
| Brand B Bean Chips Spicy Queso, High Protein and Fiber, Gluten Free, Vegan Snack, 5.5 Ounce (Pack of 6) † | chips-and -crisps | Cheddar | Navy Beans, Cayenne Pepper, Paprika Extract, Dehydrated Spices |
| Carolina Reaper Spicy Peanut Brittle | candy -brittle | Carolina Reaper Spicy | Peanuts, Sugar, Carolina Reaper |

† We mask the brand of the products to avoid revealing sensitive information.

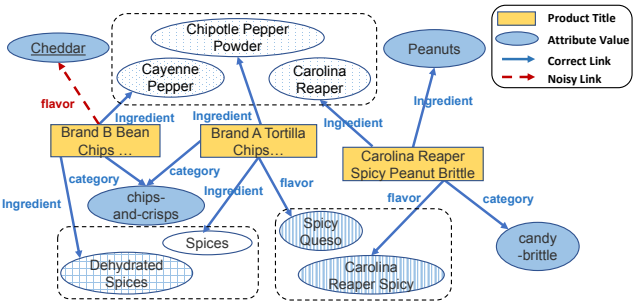


**Figure 1: An example PG and its corresponding product catalog data. We underline the incorrect attribute value in the table whose ground truth value is given in its product title. Attribute values with similar semantic meanings are filled with the same pattern and gathering together with a dotted frame.**

A PG is a knowledge graph (KG) that describes product attribute values. It is constructed based on product catalog data (Fig. 1 shows an example). In a PG, each product is associated with multiple attributes such as product brand, product category, and other information related to product properties such as flavor and ingredient. Different from traditional KGs, where most triples are in the form of (head entity, relation, tail entity), the majority of the triples in a PG have the form of (product, attribute, attribute value), where the attribute value is a short text, e.g., *("Brand A Tortilla Chips Spicy Queso, 6 - 2 oz bags", flavor, "Spicy Queso")*. We call such triples *attribute triples*.

A vast majority of the product catalog data are provided by individual retailers. These self-reported data inevitably contain many kinds of errors, including conflicting, erroneous, and ambiguous values. When such errors are ingested by a PG, they lead to unsatisfying performance of its downstream applications. Due to the huge

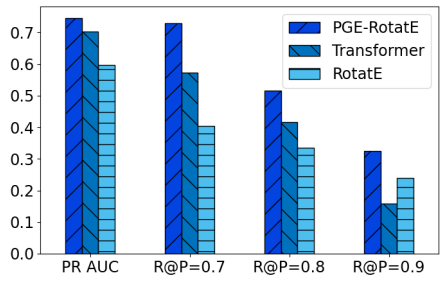


**Figure 2: PGE improves over KG embedding method RotatE by 24.7% and transformer by 4% on PR AUC in transductive setting. It also shows significant improvement on R@P metric. R@P = 0.7 shows the recall when the precision is 0.7, etc.**

volume of products in a PG, manual validation is not feasible. An automatic validation method is in urgent need.

Knowledge graph embedding (KGE) methods currently hold the state-of-the-art in learning effective representations for multi-relational graph data. It aims to learn the network structure which triples should comply. KG embedding methods have shown promising performances in error detection (i.e., determine whether a triple is correct or not) in KGs [1, 40]. For example, the PG structure in Fig. 1 indicates a strong correlation between the ingredient "pepper" and flavor "spicy" because they are connected through multiple products. By verifying its consistency with the network structure, the errors like *("Brand B Bean Chips Spicy Queso, High Protein and Fiber, Gluten Free, Vegan Snack, 5.5 Ounce (Pack of 6)", flavor, "Cheddar")* can be easily identified. Unfortunately, the existing KG embedding methods cannot be directly used to detect errors in a PG because of the following challenges.

**C1: PG contains rich textual information.** Products in a PG are often described by short texts like their titles and descriptions that contain rich information about their attributes. For example, the product title "Brand A Tortilla Chips Spicy Queso, 6 - 2 oz bags" covers multiple attributes, including brand, product category, flavor, and size. We can easily verify the correctness of these attributes against the product title. In addition, the attribute values in PG are free texts. Thus the traditional way of mapping entity ids to their embeddings is no longer appropriate. As shown in Fig. 1, when the attribute values "Chipotle Pepper Powder" and "Carolina Reaper" (a kind of pepper) are modeled as two independent entities using their ids, the strong conceptual correlation between the ingredient "pepper" and the flavor "spicy" is lost. Although several recent publications [37, 38] tried to exploit the rich textual information in KGs, the network structure and text information were not jointly encoded into a unified representation. For example, text-based representation and structure-based representation were learned by separate loss functions and integrated into one joint representation by a linear combination [1, 38].

**C2: PG contains a large number of unseen attribute values.** The flexibility of textual attribute values also makes handling "unseen attribute values" challenging. In the example as shown in Fig. 1, we can learn the representation of "Chipotle Pepper Powder" during training, but an unobserved attribute value with similar semantic meaning, such as "Chipotle Pepper" might be given for validation. Conventional KG embedding models cannot deal with this inductive setting because they have no representations for the entities outside of KGs.

**Table 1: Capabilities of different methods.**

| Methods | Modeling graph structure | Modeling textual data | Noise-aware |
|---|---|---|---|
| Structure based KG embedding [9, 31, 32, 41] | ✓ | | |
| Text and KG joint embedding [1, 37, 38] | ✓ | ✓ | |
| Noise-aware KG embedding [39] | ✓ | | ✓ |
| PGE | ✓ | ✓ | ✓ |

**C3: Existing noisy data in PG make it hard to learn a reliable embedding model.** Getting a reliable embedding model for error detection in a PG requires clean data for training. However, noise widely existing in a PG can mislead the embedding model to learn the wrong structure information, which may severely downgrade its performance in error detection.

No existing approach is capable of tackling all aforementioned challenges, as shown in Table 1. Therefore, in this paper, we aim to answer this challenging research question: *how to generate embeddings for a text-rich, error-prone knowledge graph to facilitate error detection?* We present a novel embedding learning framework, robust **P**roduct **G**raph **E**mbedding (**PGE**), to learn effective embeddings for such knowledge graphs. There are two key underlying ideas for our framework. First, our embeddings seamlessly combine the signals from the textual information of attribute triples, and the structural information in the knowledge graph. We do this by applying a CNN encoder to learn text-based representations for product titles and attribute values, and then integrating these text-based representations into the triplet structure to capture the underlying patterns in the knowledge graph. Second, we present a noise-aware loss function to prevent noisy triples in the PG from misguiding the embeddings during training. For each positive instance in the training data, our model predicts the correctness of the triple according to its consistency with the rest of the triples in the KG, and downweights an instance when the confidence of its correctness is low. As shown in Table 1, PGE is able to model both textual evidence and graph structure, and is robust to noise.

Our proposed model is generic and scalable. First, it applies not only on the product domain, but also excel in other domains such as on Freebase KG, as we show in our experiments. Second, through careful choices of the deep learning models, our model can be trained on KGs with millions of nodes within a few hours, and are robust to noises and unseen values that are inherent in real data. In summary, this paper makes the following contributions.

- We propose an end-to-end noise-tolerant embedding learning framework, PGE, to jointly leverage both text information and graph structure in PG to learn embeddings for error detection.
- We propose a novel noise-aware mechanism to incorporate triple confidence into PGE model to detect noise while learning knowledge representations simultaneously.

- We evaluate PGE on a real-world PG w. millions of nodes generated from public Amazon website and show that we are able to improve over state-of-the-art methods on average by 18% on PR AUC in transductive setting as summarized in Figure 2.

## 2 PRELIMINARIES AND PROBLEM DEFINITION

We first formally define two important concepts: attribute triples and product graph.

Definition 1. *Attribute triples*

An attribute triple can be represented as $(t, a, v)$, where its subject entity $t$ is a product sold on Amazon (e.g., a product with title "Brand A Tortilla Chips Spicy Queso, 6 - 2 oz bags"), its object entity $v$ is an attribute value (e.g., "spicy queso"), and $a$ is an attribute to connect $t$ and $v$ (e.g., flavor). Both $t$ and $v$ are represented as unstructured short texts. An attribute triple $(t, a, v)$ is *incorrect* if its attribute value $v$ does not correctly describe the product $t$. For example, *("Brand B Bean Chips Spicy Queso, High Protein and Fiber, Gluten Free, Vegan Snack, 5.5 Ounce (Pack of 6)", flavor, "Cheddar")* in Fig. 1 is an incorrect attribute triple.

Definition 2. *Product Graph*

A Product graph (PG) is a KG that describes product attribute values. Formally, we represent a product graph as $\mathcal{G} = \{T, A, V, O\}$, where $T$ is a set of product titles, $A$ is a set of attributes, $V$ is a set of product attribute values, and $O$ is a set of observed triples in the PG. Note that we have open-world assumption and thus cannot predetermine the possible values of $V$. Triples in PG are attribute triples defined in Definition 1. Fig. 1 illustrates an example PG.

We can now formally define the problem of *error detection in PG* as follows:

**Given:** a product graph $\mathcal{G} = \{T, V, A, O\}$.
**Identify:** incorrect triples $\{(t, a, v)\} \subset O$.

## 3 OUR PROPOSED FRAMEWORK: PGE

In this section, we present PGE that learns the embeddings of PG entities by incorporating both the text information and the network structure of a PG to detect erroneous triples. As shown in Fig. 3, the framework includes three key components: (1) Learn text-based representations of entities from their raw text values; (2) Leverage network structure of a PG to guide the final embedding learning for error detection; (3) Introduce a noise-aware mechanism to diminish the impact of noisy triples to the representation learning.

### 3.1 Text-based Representation Learning for Entities

In a typical KG embedding learning procedure, each entity is given a unique id which is then mapped to a learnable embedding. This approach is not optimal for PG embedding learning, because product titles ($T$) and attribute values ($V$) in a PG are mostly unstructured text containing rich semantic information, thus learning entity embeddings from only their ids not only creates unnecessary degrees of freedom, but also discards their underlying semantic connections. For instance, the embeddings of product titles "Brand A Tortilla Chips Spicy Queso, 6 - 2 oz bags" and "Brand B Bean Chips Spicy Queso, High Protein and Fiber, Gluten Free, Vegan Snack, 5.5 Ounce (Pack of 6)" should be close to each other because they are semantically similar. There are several methods, such as convolutional neural network (CNN)-based methods and Transformer-based methods (e.g., BERT), that could be leveraged to learn the representations of product titles ($T$) and attribute values ($V$) in order to capture their semantic similarities. We present scalability analysis of both text encoders in Section 4.6. Due to the huge number of products contained in PGs, we pick the CNN architecture for its good scalability as well as effectiveness on many natural language processing tasks [28]. As shown in Figure 4, the CNN encoder takes the raw text of a product title or an attribute value as the input and output its text-based representation. The first layer in the encoder transforms every word in the sequence into its respective embedding (initialized with word2vec [25]). The word embeddings then pass through three 1-d shallow CNNs with different filter sizes, which create three feature maps. Here we use different filter sizes to capture local semantic information from different text spans. The final text-based representation of an entity is the concatenated feature maps learned by all CNNs.

### 3.2 Leverage Graph Structure to Guide Embedding Learning

Manually labeled data are costly to obtain given the huge number of products in a PG. Fortunately, the rich structure information of a PG bridges the gap between the difficulties in obtaining labeled data and the necessity of supervision to detect errors. Although several recent papers have proposed to combine the text and structure information for KG representation learning, most of them [1, 38] learn two independent representations with separate loss functions and then integrate them with a linear combination. Such solution cannot generate a desired unified representation. To address this issue, we propose to learn the embeddings of entities and relations end to end, encoding the network structure that triples should obey on top of their text-based representations.

As shown in Fig. 3, we introduce a fully-connected neural network layer to transform a text-based representation into its final representation to encode the network structure of a PG. Boldfaced $\mathbf{t}$, $\mathbf{a}$, $\mathbf{v}$ denote the final embedding vector of product title $t$, attribute $a$, attribute value $v$, respectively. Since the number of attributes in a PG is small and well-defined comparing to titles and attribute values, we use randomly initialized learnable vectors to represent relations instead of CNN encoders. To capture the network structure of PG, we define the objective function by maximizing the joint probability of the observed triples given the embeddings of both entities and relations. In particular, we assume all triples are conditionally independent given the corresponding embeddings. Then the joint distribution of all the triples is defined as:

$$P(O) = \prod_{(t,a,v)\in O} P\Big((t, a, v)|\{\mathbf{t}\}, \{\mathbf{a}\}, \{\mathbf{v}\}\Big). \tag{1}$$

Since our goal is to detect the incorrect attribute value $v$ in a triple $(t, a, v)$, we optimize $P(v|t, a, \{\mathbf{t}\}, \{\mathbf{a}\}, \{\mathbf{v}\})$ instead of $P(t, a, v)|\{\mathbf{t}\}, \{\mathbf{a}\}, \{\mathbf{v}\})$, which can be formalized as follows:

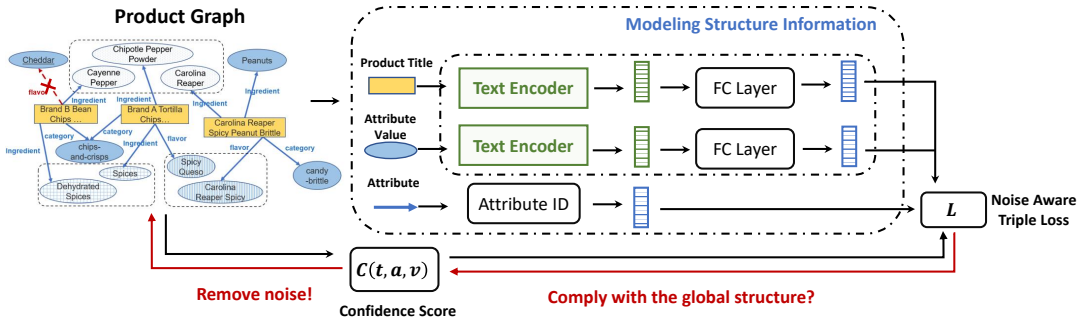


**Figure 3: Illustration of the end-to-end PGE framework. The embedding vectors in green are text-based entities representations learned from text descriptions, while the embedding vectors in blue are the final entity embeddings learned under the guidance of the PG network structure. The arrows in red illustrate how the noise-aware mechanism removes noises in PG.**

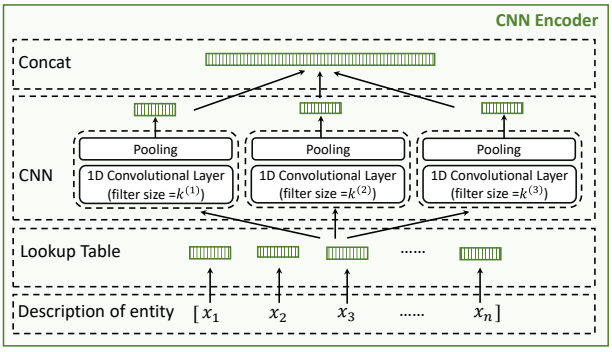


**Figure 4: CNN-based text encoder.**

$$P(v|t,a,\{\mathbf{t}\},\{\mathbf{a}\},\{\mathbf{v}\}) = \frac{\exp\Big(f_a(t,v)\Big)}{\sum_{v' \in V} \exp\Big(f_a(t,v')\Big)} \tag{2}$$

where $f_a(t,v)$ can be defined by any KG embedding scoring functions. For example, in TransE, $f_a(t,v) = \gamma - \|\mathbf{t}+\mathbf{a}-\mathbf{v}\|_1^2$, where $\mathbf{t},\mathbf{a},\mathbf{v} \in \mathbb{R}^d$ and $\gamma$ is a fixed margin. In particular, a higher $f_a(t,v)$ usually indicates that the triple $(t,a,v)$ is more plausible. Due to the large number of attribute values $|V|$ involved in a PG, it is impractical to directly compute the softmax functions. Therefore, we adopt negative sampling [26] as computationally efficient approximation instead and reformulate the objective function as follows:

$$\sum_{(t,a,v)\in O} \Big[ -\log\sigma\Big(f_a(t,v)\Big) - \frac{1}{|\mathcal{N}(t,a,v)|} \sum_{(t,a,v')\in\mathcal{N}(t,a,v)} \log\sigma\Big(-f_a(t,v')\Big)\Big] \tag{3}$$

where $\sigma$ is the standard sigmoid function, $O$ represents the observed facts in PG, $\mathcal{N}(t,a,v)$ is a set of negative samples for an attribute triple $(t,a,v)$. More specifically, for each observed triple $(t,a,v)$ we sample a set of negative samples $\mathcal{N}(t,a,v) \subset \{(t,a,v')|v' \in V\}$ by replacing the attribute value $v$ with a random value from $V$.

## 3.3 Noise-aware Mechanism

The objective function in Eq. (3) indiscriminately minimizes the scores of all facts in PG without taking their trustworthiness into consideration. As a result, noisy facts can mislead the embedding model to learn wrong structure information, thus harm the performance of embeddings in error detection.

To address this issue, we propose a novel noise-aware mechanism to reduce the impact of noisy triples on the representation learning process. Knowledge representations are learned to ensure global consistency with all triples in PG. Correct triples are inherently consistent, which can jointly represent the global network structure of PG; noisy triples usually conflict with these global network structures. Consequently, by forcing consistency between correct triples and noises, performance is unnecessarily sacrificed. The main idea of the noise-aware mechanism is to explicitly allow the model to identify and "markdown" a small set of incorrect triples during training and reduce their impact on the loss function.

More specifically, we introduce a binary learnable confidence score, $C(t,a,v)$, for every triple $(t,a,v)$ in a PG to indicate whether the fact is true or false. $C(t,a,v) = 1$ indicates the triple is correct and 0 otherwise. Associating confidence scores with triples in a PG actively downweight potential noises in the PG. The objective function of the noise-aware PGE model is defined as follows.

$$\begin{aligned}\mathcal{L} = &\sum_{(t,a,v)\in O} C(t,a,v)\Big[-\log\sigma\Big(f_a(t,v)\Big) \\ &- \frac{1}{|\mathcal{N}(t,a,v)|}\sum_{(t,a,v')\in\mathcal{N}(t,a,v)} \log\sigma\Big(-f_a(t,v')\Big)\Big] \\ &+ \alpha \sum_{(t,a,v)\in O}\Big(1-C(t,a,v)\Big) \\ &s.t., C(t,a,v) \in \{0,1\}\end{aligned} \tag{4}$$

where $C(t,a,v)$ is the binary confidence score assigned to a triple $(t,a,v)$ in a PG, and $\alpha\sum_{(t,a,v)\in O}\Big(1-C(t,a,v)\Big)$ is a regularization term imposed on confidence scores to control their sparsity. The problem in Eq.( 4) is difficult to solve due to the boolean constraint on $C(t,a,v)$. Following the common relaxation technique in [34],

the boolean constraint on $C(t,a,v)$ can be relaxed as:

$$C(t,a,v)^2 + \Big(1 - C(t,a,v)\Big)^2 = 1, \tag{5}$$

since minimizing $1 - C(t,a,v)^2 - \Big(1 - C(t,a,v)\Big)^2$ polarizes $C(t,a,v)$. Therefore, we rewrite the Eq. (4) as:

$$\begin{aligned}\mathcal{L} = & \sum_{(t,a,v)\in O} C(t,a,v)\Big[ -\log\sigma\Big(f_a(t,v)\Big) \\ & - \frac{1}{|\mathcal{N}(t,a,v)|} \sum_{(t,a,v')\in\mathcal{N}(t,a,v)} \log\sigma\Big(-f_a(t,v')\Big)\Big] \\ & + \alpha \sum_{(t,a,v)\in O} \Big(1 - C(t,a,v)\Big) \\ & + \beta \sum_{(t,a,v)\in O} \Big(1 - C(t,a,v)^2 - \Big(1 - C(t,a,v)\Big)^2\Big).\end{aligned} \tag{6}$$

# 4 EXPERIMENTS

## 4.1 Dataset

We evaluate our PGE on two datasets: one real-world e-commerce dataset collected from publicly available Amazon webpages, and one widely used benchmark dataset FB15K-237. Table 2 summarizes the statistics of both datasets.

**Amazon Dataset:** To evaluate PGE on real-world e-commerce dataset, we construct a product graph with the product data obtained from public Amazon website. Each product in the Amazon dataset is associated with multiple attributes, such as product title, brand and flavor, whose values are short texts. As shown in Table 2, the Amazon dataset contains 750,000 products associated with 27 structured attributes and 5 million triples. To avoid bias, we sampled products from 325 product categories across different domains, such as food, beauty and drug. To prepare labeled test data, we asked Amazon Mechanical Turk (MTurk) workers to manually label the correctness of two attributes, including *flavor* and *scent*, based on corresponding product profiles. Each data point is annotated by three Amazon Mechanical Turk workers and the final label is decided by majority voting. Among 5,782 test triples, 2,930 are labeled as incorrect and 3,304 are labeled as correct.

**FB15K-237:** The FB15K dataset is the most commonly used benchmark knowledge graph dataset [9]. It contains knowledge graph relation triples and textual mentions of Freebase entity pairs. FB15K-237 is a variant of FB15K dataset where inverse relations are removed to avoid information leakage problem in test dataset. The FB15K-237 datasets benefit from human curation that results in highly reliable facts. We add 10% noisy triples to the data set by randomly sample 10% triples and substituting the original head or tail entity with a randomly selected entity.

## 4.2 Experimental Setting

Our goal is to identify incorrect attribute values of a product, which can be formally defined as a triple classification problem in PG. We choose a threshold $\theta$ based on the best classification accuracies on the validation dataset, then classify a triple $(t,a,v)$ as correct if its score $f_a(t,v) > \theta$, otherwise incorrect. We apply the same settings to all baseline methods to ensure a fair comparison. We evaluate two versions of our model by incorporating TransE [9] and RotatE [31] as the score function, respectively.

**Evaluation Metric.** We adopt the area under the Precision-Recall curve (PR AUC) and Recall at Precision=X (R@P=X) to evaluate the performance of the models. To be more specific, PR AUC is defined as the area under the precision-recall curve, which is widely used to evaluate the ranked retrieval results. R@P is defined as the recall value at a given precision, which aims to evaluate the model performance when a specific precision requirement needs to be satisfied. For example, R@R = 0.7 shows the recall when the precision is 0.7.

**Compared Methods.** We evaluate PGE against state-of-the-art (SOTA) algorithms, including (1) NLP-based method (LSTM, Transformer [33]); (2) structure based KG embedding (TransE [9], DistMult [41], ComplEx [32], RotatE [31]); (3) text and KG joint embedding (e.g., DKRL [38], SSP [37]); and (4) noise-aware KG embedding (CKRL [39]). We choose CNN and BERT as the text encoders of PGE. Since BERT cannot handle Amazon dataset due to scalability issues, only the results of CNN is reported in Section 4.3 and Section 4.4. We present scalability analysis of both text encoders in Section 4.6. We also include the approach "Union of Transformer and PGE" to show how PGE complement Transformer. To combine Transformer and PGE for error detection, the approach "Union of Transformer and PGE" re-ranks the test triples by jointly considering the ranking given by the Transformer and PGE. For example, given a test triple $(h,r,t)$, suppose Transformer rank it as i while PGE rank it as j. Then the average ranking of triple $(h,r,t)$ is $R_{avg}^{(h,r,t)} = (1/i + 1/j)/2$. Based on $R_{avg}^{(h,r,t)}$, "Union of Transformer and PGE" re-ranks the test triples. Smaller $R_{avg}^{(h,r,t)}$ results in higher ranking assigned by "Union of Transformer and PGE". In addition to "Union of Transformer and PGE", we also include a strong ensemble method - RotatE+ to enrich knowledge graph with information extraction technique. In particular, RotatE+ first applies OpenTag [20, 43], the SOTA information extraction toolkit developed by Amazon Product Graph Team, to extract all relevant attributes from product title and product description to enrich the PG, then applies KG embedding method RotatE on the enriched KG to detect the error.

**Setup Details.** In data preprocessing, we remove all stop words from raw texts and map words to 300-dimensional word2vec vectors trained with GoogleNews. We adopt the Adam [23] optimizer with learning rate among $\{0.0001, 0.0002, 0.0005\}$ following [31], and margin $\gamma$ among $\{12.0, 24.0\}$. For the CNN encoder, we try different filter sizes among $\{1,2,3,4\}$ for different CNNs. To fairly compare with different baseline methods, we set the parameters for all baseline methods by a grid search strategy. The best results of baseline methods are used to compare with PGE.

## 4.3 Transductive Setting

Transductive setting focuses on the situation where all attribute values in the test triples have been observed in the training stage. To compare different algorithms on the triple classification task, we require each method to predict the correctness of triples in the test dataset. Table 3 shows the comparison results. Here are several interesting observations: (1) PGE consistently outperforms KG embedding models as well as CKRL in all cases with significant

**Table 2: Data statistics**

| Dataset | #Relations | #Entities | #Products | #Attributed values | #Train | #Valid | #Test |
|---|---|---|---|---|---|---|---|
| Amazon Dataset | 27 | 1,017,374 | 750,000 | 267,374 | 4,989,375 | 6,924 | 5,782 |
| FB15K-237 | 234 | 13,714 | - | - | 67,894 | 2,750 | 3,042 |

performance gain (improving by 24% - 30% on PR AUC), which ascribes to the utilization of textual information associated with entities; (2) PGE also obtains better performance than NLP-based approaches as they cannot leverage graph structure information in KGs. In particular, NLP-based methods show the worst performance on the FB15k-237 dataset while the second best performance on Amazon dataset. The major reason is that FB15k-237 contains much richer graph information compared to the Amazon dataset (i.e., there are 27 attributes in Amazon dataset while 234 relations in FB15k-237). Therefore, graph structure plays a more critical role in error detection task in FB15k-237; (3) PGE shows better performance compared to DKRL and SSP. The major reason is that DKRL and SSP learn the structural representations and the textual representations by separate functions.

## 4.4 Inductive Setting

Inductive setting focuses on the situation where attribute values in the test triples are not presented in a PG, which is a common scenario for PG error detection. Existing KG embedding models are not effective in dealing with this situation because they cannot generate representations for the entities outside of KGs due to missing ids. Therefore, we do not include them as baselines in this subsection. Unlike the KG embedding methods, which map entities to their embeddings using ids, our proposed PGE learns embeddings of entities based on their text-based representations and thus can naturally handle the inductive setting.

To prepare an inductive setting, we filter the training set by excluding any triples that share entities with the selected test triples, so that the training and the testing use disjoint sets of entities. We report the results on R@P=0.6, R@P=0.7, R@P=0.8 in Table 4. We observe that: (1) All methods perform worse in the inductive setting without exception, which indicates that inductive setting is indeed more challenging; (2) NLP-based methods perform the best among all methods. The major reason is that language naturally has strong transferring ability while PGE still relies on the graph structure to make the prediction. Although text encode can transferring information among entities, it doesn't help to predict a never seen graph structure; (3) Although NLP-based methods perform better than PGE on the Amazon dataset, the best results are given by the union of Transformer and PGE (improving by 9% on R@P=0.9 compared with Transformer), showing that PGE can learn the undetected error by Transformer ; (4) Although PGE cannot leverage textual information as well as Transformer (because CNN is less powerful compared with Transformer in capturing the semantic information. Not to mention we employ shallow CNN as text encode due to scalability issues), it still achieves comparable result on the Amazon dataset. Moreover, they achieve the SOTA on FB15k-237, which further validates the strong ability of PGE in detecting errors in a KG with rich textual information; (5) DKRL and SSP perform the worst among all methods, which again demonstrates their weakness.

## 4.5 Validity of Noise-aware Mechanism

**Validity of Confidence Scores with Different Injected Noises.** To evaluate the benefit of including confidence scores $C(t, a, v)$ in the noise-aware mechanism, shown in Eq. (6) , we evaluate *PGE(CNN)-RotatE* on the Amazon dataset with two different kinds of injected noises. First, we inject human-labeled correct triples and incorrect triples into the training data. Confidence scores are learned to determine the correctness of these injected labeled triples. The distribution of confidence scores are shown in Fig. 5 (a). Second, we inject artificial noises into the training data. We substitute attribute values of existing triples in the Amazon dataset with a random value to generate these artificial noises. Confidence scores are learned to distinguish artificial noises from the original triples. Fig. 5 (b) shows the distribution of confidence scores. The red bars represent the distribution of confidence scores for human-labeled incorrect triples (or injected artificial noises) while the blue bars represent the distribution of confidence scores for human-labeled correct triples (or triples in the original Amazon dataset). We observe that real-world noises are more difficult to identify compared to artificial noises. Despite the difficulty in detecting the real-world error, confidence scores of human-labeled correct triples are mainly over 0.5, validating the promising capability of the confidence scores to distinguish noises in PG. In addition, we observe that 1% triples in the original Amazon dataset have also been identified as noises in Fig. 5 (b). We have verified that most of these triples are indeed noisy triples in the original Amazon dataset.

**Overall Impact of Noise-aware Mechanism.** To further validate the overall benefits brought by noise-aware mechanism, we also evaluate *PGE(CNN)-RotatE* without noise-aware mechanism on Amazon dataset used in Section 4.3. Figure 6 presents the comparison results. We observe that the noise-aware mechanism brings significant performance gain: PGE(CNN)-RotatE with noise-aware mechanism increases the PR AUC of PGE(CNN)-RotatE without noise-aware mechanism from 0.734 to 0.747 and increases R@P=0.9 from 0.289 to 0.325.

## 4.6 Scalability Analysis

To demonstrate the scalability of PGE, we present the training time of PGE on Amazon dataset of different sizes in Table 5. We vary the sample ratio among $\{0.1, 0.3, 0.5, 0.7, 1\}$ to select only a portion of triples in Amazon dataset to construct PG of different sizes. Two text encoders, CNN-based text encoder and BERT-based text encoder, are leveraged to learn entity representations. In particular, BERT-based text encoder takes the raw text of product titles or attribute values as input. The first token of input is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the text-based representation of entities. We observe that *PGE(BERT)-RotatE* cannot be applied to Amazon dataset due to the scalability issue. It takes near 2 days for 10% data and over 3 days for 30% data. Therefore, we focus on CNN in this

**Table 3: Results of error detection under the transductive setting. The numbers in bold represent the best performance among all methods while the numbers underlined represent the second best. Evaluation of PGE on the Amazon dataset shows that PGE is able to improve over the SOTA methods on average by 18% on PR AUC.**

| Categories | Method | Amazon Dataset | | | | | FB15k-237 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PR AUC | R@P=0.7 | R@P=0.8 | R@P=0.9 | Time (hours) | PR AUC | R@P=0.7 | R@P=0.8 | R@P=0.9 | Time (hours) |
| NLP-based methods | LSTM | 0.704 | 0.572 | 0.416 | 0.159 | 16.32 | 0.626 | 0.595 | 0.445 | 0.239 | 0.43 |
| | Transformer [33] | 0.719 | 0.601 | 0.427 | 0.194 | 79.46 | 0.648 | 0.649 | 0.503 | 0.245 | 12.82 |
| Structured based KG embedding | TransE [9] | 0.584 | 0.390 | 0.308 | 0.213 | 20.57 | 0.772 | 0.793 | 0.737 | 0.685 | 0.58 |
| | DistMult [41] | 0.573 | 0.362 | 0.291 | 0.197 | 32.86 | 0.819 | 0.872 | 0.813 | 0.751 | 4.12 |
| | ComplEx [32] | 0.579 | 0.373 | 0.310 | 0.207 | 36.31 | 0.781 | 0.814 | 0.759 | 0.712 | 5.16 |
| | RotatE [31] | 0.597 | 0.405 | 0.336 | 0.239 | 35.11 | 0.824 | 0.875 | 0.823 | 0.766 | 5.33 |
| | RotatE+‡ | 0.611 | 0.423 | 0.369 | 0.221 | 36.79 | - | - | - | - | - |
| Text and KG joint embedding | DKRL [38] | 0.693 | 0.552 | 0.408 | 0.246 | 45.38 | 0.909 | 0.945 | 0.901 | 0.868 | 7.25 |
| | SSP [37]† | - | - | - | - | - | 0.927 | 0.951 | 0.915 | 0.882 | - |
| Noise-aware KG embedding | CKRL [39] | 0.586 | 0.392 | 0.304 | 0.217 | 21.16 | 0.768 | 0.725 | 0.672 | 0.627 | 0.62 |
| Our Proposed model | PGE(CNN)-TransE | 0.738 | 0.690 | 0.436 | 0.267 | 23.12 | **0.990** | **0.997** | **0.995** | **0.986** | 0.67 |
| | PGE(CNN)-RotatE | 0.745 | 0.729 | **0.516** | 0.325 | 39.41 | **0.990** | **0.997** | 0.993 | 0.983 | 5.71 |
| Union of Transformer and PGE(CNN)-RotatE | | **0.751** | **0.747** | 0.509 | **0.349** | - | 0.938 | 0.958 | 0.911 | 0.893 | - |

† Since SSP cannot handle Amazon dataset due to scalability issues, only the results on the FB15K-237 is reported.

‡ RotatE+ first applies OpenTag [20, 43], an information extraction toolkit developed by Amazon Product Graph Team, to extract all relevant attributes from product title and product description to enrich the product graph, then apply RotatE [31] on the enriched KG to detect the error.

**Table 4: Results of error detection under the inductive setting. The bold numbers represent the best performances among all methods while the underlined numbers represent the second best. We observe that PGE achieves the SOTA on the structure-rich FB15k-237 data set. The best results on the Amazon dataset are given by the union of the Transformer model and PGE, showing that although PGE does not perform as well as NLP-based methods on the Amazon dataset, it complements Transformer for its strong ability in capturing graph structure.**

| Categories | Method | Amazon Dataset | | | | FB15k-237 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | PR AUC | R@P=0.6 | R@P=0.7 | R@P=0.8 | PR AUC | R@P=0.6 | R@P=0.7 | R@P=0.8 |
| NLP-based methods | LSTM | 0.626 | 0.756 | 0.476 | 0.340 | 0.581 | 0.717 | 0.436 | 0.204 |
| | Transformer [33] | 0.643 | 0.771 | 0.495 | 0.354 | 0.603 | 0.748 | 0.453 | 0.238 |
| Text and KG joint embedding | DKRL [38] | 0.552 | 0.593 | 0.252 | 0.068 | 0.698 | 0.790 | 0.638 | 0.415 |
| | SSP [37]† | - | - | - | - | 0.716 | 0.807 | 0.654 | 0.419 |
| Our Proposed model | PGE(CNN)-TransE | 0.585 | 0.730 | 0.412 | 0.197 | 0.787 | 0.871 | 0.724 | 0.674 |
| | PGE(CNN)-RotatE | 0.596 | 0.741 | 0.437 | 0.228 | **0.836** | 0.919 | **0.845** | **0.753** |
| Union of Transformer and PGE(CNN)-RotatE | | **0.649** | **0.779** | **0.512** | **0.386** | 0.833 | **0.923** | 0.837 | 0.743 |

† Since SSP cannot handle Amazon dataset due to scalability issues, only the results on the FB15K-237 is reported.

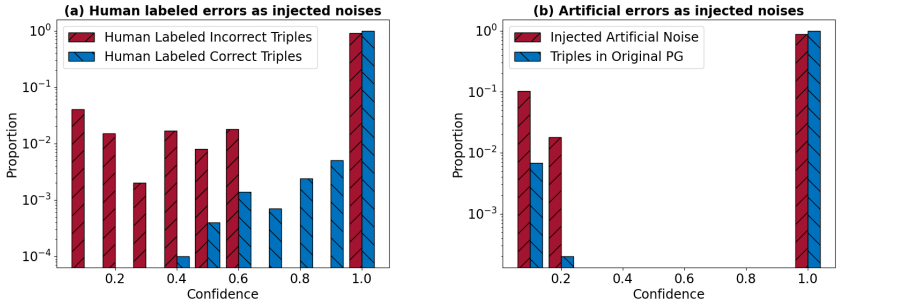


**Figure 5: Distribution of confidence scores learned by PGE(CNN)-RotatE on the Amazon dataset with different injected noises.**

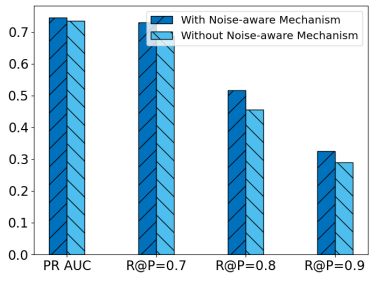


**Figure 6: PGE(CNN)-RotatE with v.s. without noise-aware mechanism on noisy Amazon dataset.**

paper. We observe that *PGE(CNN)-RotatE* scales up to large datasets with similar scalability compared to KGE model.

## 4.7 Case Study

Previous experiments have shown the promising performance of PGE in both transductive setting and inductive setting. To further demonstrate the capability of PGE in detecting real-world errors in PG, we conduct case study to give examples of identified errors in the Amazon dataset as shown in Table 6. We use PGE(CNN)-RotatE to evaluate if a triple is a correct fact. Threshold $\sigma$ is chosen based on the best classification accuracies on the validation dataset in order to classify triples. We observe that most attribute values of identified errors violate the global graph structure of PG and thus can be classified as errors. For example, product 2,3, and 4 in Table 6 are not groceries thus should not have the attribute "flavor".

**Table 5: Training Time (hours) of different methods on Amazon dataset.**

| Models | Percentage of Sampled Triples | | | | |
|---|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | 0.7 | 1 |
| RotatE | 3.22 | 10.77 | 17.63 | 24.86 | 35.11 |
| PGE(CNN)-RotatE | 4.07 | 11.95 | 19.44 | 27.62 | 39.41 |
| PGE(BERT)-RotatE | 45.21 | > 3 day | > 3 day | > 3 day | > 3 day |

**Table 6: Identified errors on Amazon dataset.**

| Product | Attribute | Attribute Value |
|---|---|---|
| Pure Mint Shampoo and Hair Conditioner for Women and Men - 10 oz | scent | mint shampoo and conditioner set |
| Brand A Foot Brush and Pumice (Pack of 4) † | flavor | bamboo |
| Brand B Sweet BBQ Rub 11.2 oz† | flavor | sweet |
| Hassle Free Storage Pop-Up Mesh Laundry Hamper (Aqua) | flavor | octopus |
| Brand C Organics Conditioner, Tea Tree Oil & Blue Cypress, † 12 Ounce (Pack of 3) | scent | conditioner tea tree oil and blue cypress |

† We mask the brand of the products to avoid revealing sensitive information.

Although the attribute values of product 1 and 5 include commonly observed phrases to describe the scent, the word "conditioner" in the attribute values makes them no longer correct attribute values of "scent". This observation shows that PGE not only leverages the graph structure of PG to detect noise (e.g., example 2,3,4) but also show sensitivity to subtle differences of language.

## 5 RELATED WORK

**Error Detection in Knowledge Graph.** Most KG noise detection process is carried out when constructing KGs, such as Freebase, Google Knowledge Graph, Walmart product graph, YAGO, NELL, and Wikipedia [3, 8, 19, 27, 30]. Despite the efforts during KG constructions, errors are widely observed in existing KGs. A recent open IE model on the benchmark achieves only 24% precision when the recall is 67% [29] and the estimated precision of NELL is only 74% [10]. To detect errors for an existing KG, most existing methods explore additional rules [5–7, 11, 12, 15–18, 22]. Considering all kinds of errors that could be made in the real world, it is unrealistic to identify all required rules to cover all possible cases. In contrast, our proposed method employs KG embedding model to automatically learn the correlation of entities, which could be considered as fuzzy rules to guide value cleaning in KGs. More recently, detecting noises while learning knowledge representations simultaneously becomes a hot topic. A confidence-aware framework CKRL [39] is proposed to incorporate triple confidence into KG embedding models to learn noise-aware KG representations. However, the confidence of triples are easily affected by model bias (i.e., improper order of triples in training sets may even amplify the impact of noises). In addition, it ignores the rich semantic information in KGs, which is strong evidence to judge triple quality. In this paper, we propose a noise-aware KG embedding learning method, which can utilize rich semantic information to identify noises, which conflict with the global network structures.

**Knowledge Graph Embedding.** Knowledge Graph Embedding (KGE) aims to capture the similarity of entities by projecting entities and relations into continuous low-dimensional vectors. Scoring functions, which measure the plausibility of triples in KGs, are the crux of KGE models. Representative KGE algorithms include TransE [9], TransH [36], TransR [24], DistMult [41], ComplEx [32], SimplE [21] and RotatE [31], which differ from each other with different scoring functions.

**Text and Knowledge Graph Joint Embeddings.** In recent years, several attempts have been made to improve the knowledge representation by exploiting entity descriptions as additional information [1, 37, 40]. However, the combination of the structural and textual representations is not well studied in these methods, in which two representations are learned by separate loss function or aligned only on the word-level. As one of the most representative works, DKRL [38] separates the objective function into two energy functions (i.e. one for structure and one for description) and integrates these two representations into a joint one by a linear combination. Works proposed in [36] and [44] align the entity name with its Wikipedia anchor on word level, which may lose some semantic information on the phrase or sentence level. SSP [37] requires the topic model to learn pre-trained semantic vector of entities separately. Due to the rapid growth of pre-trained language representation models (PLM), several works are proposed to encode textual entity descriptions with a PLM as their embeddings. For example, KEPLER [35] proposes to encode textual entity descriptions with BERT as their embeddings, and then jointly optimize the KGE and language modeling objectives. BLP [13] trains PLM and KG in an end-to-end manner. Since the language modeling objective of PLM suffer from high computational cost and require a large corpus for training, it is time consuming to apply these methods to large scale KGs. In this paper, we propose an end-to-end method to jointly leverage both text information and graph structure for KG embedding learning in an efficient way.

## 6 CONCLUSION

In this paper, we propose a novel end-to-end noise-aware embedding learning framework, PGE, to learn embeddings on top of text-based representations of entities for error detection in PG. Experiment results on a real-world product graph show that PGE improves over state-of-the-art methods on average by 18% on PR AUC in transductive setting. Although this paper focuses on the product domain, we also show in our experiments that, the same techniques excel in other domains with textual information and noises. As the next step, we would investigate more efficient Transformer architecture to improve text encoder strength and efficiency of PGE. BERT-based text encoder is difficult to scale to large KG due to its full attention mechanism. To reduce the computation complexity of BERT-based text encoder, we can to extend the ideas of [4, 42] to allow sparse self-attention to tokens. In addition, we can leverage additional information to improve the learned entity representations. For example, we could better capture the similarity among products by leveraging the hierarchical structure of product data or by leveraging the user behavior data.

# REFERENCES


[1] Bo An, Bo Chen, Xianpei Han, and Le Sun. 2018. Accurate text-enhanced knowledge graph representation learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 745–755.

[2] Vito Walter Anelli, Pierpaolo Basile, Derek Bridge, Tommaso Di Noia, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Markus Zanker. 2018. Knowledge-aware and conversational recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 521–522.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 722–735.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[5] George Beskales, Ihab F Ilyas, and Lukasz Golab. 2010. Sampling the repairs of functional dependency violations under hard constraints. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 197–207.

[6] George Beskales, Ihab F Ilyas, Lukasz Golab, and Artur Galiullin. 2013. On the relative trust between inconsistent data and inaccurate constraints. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 541–552.

[7] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *2007 IEEE 23rd international conference on data engineering*. IEEE, 746–755.

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1247–1250.

[9] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24.

[11] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 458–469.

[12] Alvaro Cortés-Calabuig and Jan Paredaens. 2012. Semantics of Constraints in RDFS.. In *AMW*. Citeseer, 75–90.

[13] Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive Entity Representations from Text via Link Prediction. In *Proceedings of the Web Conference 2021*. 798–808.

[14] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2724–2734.

[15] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems (TODS)* 33, 2 (2008), 1–48.

[16] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *Proceedings of the 2016 International Conference on Management of Data*. 1843–1857.

[17] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *Proceedings of the VLDB Endowment* 6, 9 (2013), 625–636.

[18] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*. 829–846.

[19] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism detection in wikidata. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 327–336.

[20] Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. Txtract: Taxonomy-aware knowledge extraction for thousands of product categories. *arXiv preprint arXiv:2004.13852* (2020).

[21] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*. 4284–4295.

[22] Zuhair Khayyat, Ihab F Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. 2015. Bigdansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1215–1230.

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[27] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.

[28] A Rakhlin. 2016. Convolutional Neural Networks for Sentence Classification. *GitHub* (2016).

[29] Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 885–895.

[30] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

[31] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[34] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[35] X Wang, T Gao, Z Zhu, Z Liu, J Li, and J Tang. [n.d.]. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. arXiv 2019. *arXiv preprint arXiv:1911.06136* ([n. d.]).

[36] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[37] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[38] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[39] Ruobing Xie, Zhiyuan Liu, Fen Lin, and Leyu Lin. 2018. Does William Shakespeare really write Hamlet? knowledge representation learning with confidence. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[40] Jiacheng Xu, Kan Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Knowledge graph representation with jointly structural and textual encoding. *arXiv preprint arXiv:1611.08661* (2016).

[41] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[42] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big Bird: Transformers for Longer Sequences.. In *NeurIPS*.

[43] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1049–1058.

[44] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 267–272.