

---

# Type Information-Assisted Self-Supervised Knowledge Graph Denoising

---

Jiaqi Sun<sup>1,2</sup>

Yujia Zheng<sup>1</sup>

Xinshuai Dong<sup>1</sup>

Haoyue Dai<sup>1</sup>

Kun Zhang<sup>1,2</sup>

<sup>1</sup>Carnegie Mellon University

<sup>2</sup>Mohamed bin Zayed University of Artificial Intelligence

## Abstract

Knowledge graphs serve as critical resources supporting intelligent systems, but they can be noisy due to imperfect automatic generation processes. Existing approaches to noise detection often rely on external facts, logical rule constraints, or structural embeddings. These methods are often challenged by imperfect entity alignment, flexible knowledge graph construction, and overfitting on structures. In this paper, we propose to exploit the consistency between entity and relation type information for noise detection, resulting a novel self-supervised knowledge graph denoising method that avoids those problems. We formalize *type inconsistency* noise as triples that deviate from the majority with respect to type-dependent reasoning along the topological structure. Specifically, we first extract a compact representation of a given knowledge graph via an encoder that models the type dependencies of triples. Then, the decoder reconstructs the original input knowledge graph based on the compact representation. It is worth noting that, our proposal has the potential to address the problems of knowledge graph compression and completion, although this is not our focus. For the specific task of noise detection, the discrepancy between the reconstruction results and the input knowledge graph provides an opportunity for denoising, which is facilitated by the type consistency embedded in our method. Experimental validation demonstrates the effectiveness of our approach in detecting potential noise in real-world data.

## 1 INTRODUCTION

Knowledge graphs are widely used to provide expert knowledge support for intelligent systems, such as chatbots, recommendation systems, etc. [Guo et al., 2020, Ji et al., 2021, Pan et al., 2024]. A set of triples is commonly used to represent a knowledge graph, where each triple contains a head entity, a relation, and a tail entity. Despite the fact that knowledge graphs are used as expert knowledge, they can be noisy due to the imperfect automatic generation process, which often lacks strict expert supervision [Deng et al., 2023, Ma et al., 2023]. Examples of such imperfections include computational errors during the construction of knowledge graphs from text extractions and construction bias when multiple people contribute to the construction process. These problems can be further exacerbated when using multi-hop connections over knowledge graphs, which is particularly detrimental when precise retrieval is required [Guo et al., 2020, Ji et al., 2021, Chetoui et al., 2022]. In this paper, we will focus on denoising knowledge graphs by detecting potential noise in real-world data.

Before discussing related proposals, it is necessary to clarify what kind of noise we are trying to tackle in the context of knowledge graphs. Previous work on commonsense knowledge graphs defines noise as triples that are inconsistent with objective facts or logical reasoning based on those facts [Deng et al., 2023]. However, detecting such noise is often impractical because external facts are usually missing in most knowledge graph applications, and there are entity alignment challenges if the external database is assumed to be correct [Zhao et al., 2020, Zeng et al., 2021]. Some rule-based approaches take into account the incompleteness, conflict, and redundancy of knowledge graphs [Cheng et al., 2018, Belth et al., 2020], but their strict definition of noise, based on a rigorous formulation of logical rules, is likely to be incompatible with the flexible construction process of real-world knowledge graphs. Other embedding-based methods attempt to generate a confidence score for each triple based on topological consistency, while potentially suffering from overfitting in the

structures [Zhang et al., 2022, Zhang et al., 2023]. Given the above shortcomings, we try to find a more general and practical way to uncover the noisy triples, which should go beyond topological consistencies and external facts. Consider a triple, (head\_entity, relation, tail\_entity), from the NELL-995 dataset [Xiong et al., 2017]: (concept\_city\_murdoch, concept:agentcontrols, concept\_personasia\_news\_corporation). This triple does not seem sensible because Rupert Murdoch, who the triple is supposed to represent, is a businessman, which does not logically fit the entity type city. This inconsistency can be caught by noticing that the entity type city does not typically appear with the relation type agentcontrols in the data set.

Based on this discovery, in this paper, we aim to detect noise by leveraging the consistency between the type information of entities and relations. We define *type inconsistency noise*: *Given a knowledge graph, type inconsistency noise is a set of triples whose type information is inconsistent with that of the majority of existing triples or with triples derived through type-dependent reasoning.* This definition implies a reasonable assumption that most links are correct. In particular, this definition of noise does not rely on any external input, allowing us to denoise the knowledge graph in a self-supervised manner. Moreover, the type information along with the structural relations gives rise to the constraints that the legit triples should obey. These considerations form the general idea of our proposal: a type-information assisted self-supervised knowledge graph denoising approach.

In the methodology, we introduce an auto-encoder architecture to implement this idea. Specifically, we first employ an encoder that extracts a compact representation of the entire knowledge graph, considering only the type information of triples and their structural dependencies. The decoder then reconstructs the original input knowledge graph based on this compact representation. The difference between the reconstructed result and the input knowledge graph provides an opportunity for denoising, which is facilitated by the type consistency embedded in our approach. It should be noted that the proposed method has the potential to address the problems of knowledge graph compression and completion, although this is not our focus [Sachan, 2020, Zamini et al., 2022].

In summary, our contributions are as follows: **i)** We investigate type information as a direct and effective source of revealing noisy triples in knowledge graphs. **ii)** Accordingly, we design a type information-assisted self-supervised denoising approach. **iii)** We manage to reveal noisy triples directly from real data.

## 2 NOTATIONS

**Knowledge graph and type information.** A knowledge graph consists of entities and the relations between them, typically represented as a set of triples:  $K = \{(h_i, r_i, t_i)\}_{i=1}^n$ , where each triple  $(h_i, r_i, t_i)$  comprises a head entity  $h_i$ , a relation  $r_i$ , and a tail entity  $t_i$  and there are  $n$  triples in the graph. Each entity is from an entity token domain  $V$ , i.e.,  $\forall i \in z(n), h_i, t_i \in V$ . We use  $z(n) = \{1, 2, \dots, n\}$  to denote the index set. Similarly, each relation  $r_i$  belongs to a relation domain  $R$ . A knowledge graph can alternatively be represented as a three-dimensional cube, i.e.,  $\mathbf{A} \in \{0, 1\}^{|V| \times |R| \times |V|}$ , where  $\mathbf{A}[h, r, t] = 1$  if  $(h, r, t) \in K$ , otherwise  $\mathbf{A}[h, r, t] = 0$ . Both  $(h, r, t)$  and  $(h_i, r_i, t_i)$  can represent a knowledge graph triple, and the latter is preferable when the index is needed. Moreover, in this work we focus especially on the knowledge graph that comes with type information of entities, represented by a type function  $c : V \cup R \mapsto C$ . The cardinality of the entity type domain is strictly less than that of the entity domain, i.e.,  $|c[V]| < |V|$ . However, in most knowledge graphs, the type domain and the token domain are the same for relations. Therefore, for relations, the type function is an identity mapping. In this paper, a knowledge graph with entity type information is denoted as  $G = (K, c)$  or  $G = (\mathbf{A}, c)$ .

**Other notation convention.** We use uppercase letters to denote a set, and its member is marked as lowercase or indexed, i.e.,  $x$  or  $x_i$ . The cardinality of the set  $X$  is denoted by  $|X|$ . Bold capital letters denote matrices or a three-dimensional cube (i.e., a collection of matrices). Bold lowercase letters denote a vector. The entries of a given matrix are marked by using square brackets, i.e.,  $\mathbf{X}[h, r, t]$ . Functions are denoted using lowercase letters. The output of a function for a singleton input is denoted as  $f(x)$ , and  $f[X]$  denotes the case where the input and the output are set.

## 3 METHODOLOGY

### 3.1 Observations and Motivation

**Type information.** First, let us elaborate on our motivation by demonstrating the correlation between type information and triples in a knowledge graph. Here, we provide quantitative evidence to illustrate the relationship between type information and the frequency of observing the corresponding triples. Table 1 summarizes the statistics of three widely used datasets (with detailed descriptions provided in Section 5). The cardinality of the type domain is significantly smaller than that of the token domain, as introduced in Section 2. More importantly, we calculate the proportion of legitimate triple types (% LTT)—defined as the existing com-

Dataset	NELL-995	WN18RR	FB15k-237
# entities	75,492	40,943	14,541
# entity types	267	11	237
# relations (=types)	200	11	237
% LTT	0.13	21.11	0.05

Table 1: Type information of three real-world datasets.

binations of head entity type, relation (type), and tail entity type—relative to all possible combinations of entity and relation types in the entire dataset, as follows:

$$\% \text{ LTT} = \frac{|\{(c(h), c(r), c(t)); (h, r, t) \in K\}|}{|c[V]| \times |c[R]| \times |c[V]|}, \quad (1)$$

where  $c$  is the type function within a given knowledge graph  $G = (K, c)$ . As shown in the table, legitimate triple types represent only a small fraction of all possible triple type combinations. To better illustrate the concentration of type distribution, Figure 1 shows the distribution of entity types for a randomly chosen relation type in the least concentrated dataset, WN18RR. In the figure, the darker green indicates a higher frequency of the relation type occurring between the given connected entity types. The results show that the possible entity types associated with a given relation type are very limited. Such observations are general across different datasets, which can be found in the Appendix.

This observation suggests that as the order of the link increases, the type constraints become stricter. For instance, given two entities with known types, the number of possible relation type combinations in the intermediate multi-hop link becomes more centrally distributed. Consequently, if we reasonably assume that the majority of triples in a knowledge graph are correct, and therefore their triple types are also correct, we can use this concentrated distributed type information to detect noise. Specifically, we can identify noisy triples when their type information deviates from the majority, as indicated by multi-hop constraints. Given the feasibility of using type information as a guide for noise detection, we propose a method to model type dependencies over triples to unveil hidden noise in knowledge graphs.

**Compact triple set and denoising.** Since the topology of the knowledge graph allows for information dependencies between triples, to model the type dependencies of triples, we need to consider the type information of each triple and propagate these dependencies along the structure of the graph. Furthermore, reduction along these topological dependencies confirms the existence of a subset of triples on which the type information of all other triples depends. Given the above analysis, we propose to extract a compact representation of the knowledge graph, i.e., a subset of all existing triples. This compact set is to be inferred

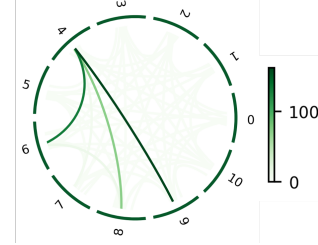


Figure 1: Entity types distribution w.r.t. a given relation in WN18RR.

from the existing triples and used to infer other legitimate triples. In this way, noisy triples are not kept in the compact set, since they are not type-dependent on most of the existing triples. Moreover, for the same reason, these noisy triples cannot be constructed from the compact representation set. Accordingly, we design a self-supervised architecture in which the encoder extracts the compact set based on type information and structural dependencies, and the decoder reconstructs the knowledge graph. The difference between the original knowledge graph triples and their reconstruction provides an avenue to detect type-inconsistency noise.

### 3.2 General Proposal

Given a knowledge graph  $G = (K, \mathbf{A}, c)$  containing  $n$  triples, i.e.,  $K = \{(h_i, r_i, t_i)\}_{i=1}^n$ , its three-dimensional cube representation  $\mathbf{A} \in \{0, 1\}^{|V| \times |R| \times |V|}$ , and the type information function of the entities, i.e.,  $c : V \mapsto C$  (here we focus on the case when the relation type domain is identical to its token domain), we want to find the most compact triples that can be used to reconstruct the original graph. To simplify the notation, we use  $\mathcal{B}$  to denote the discrete three-dimensional space:  $\{0, 1\}^{|V| \times |R| \times |V|}$ . And we use  $(h, r, t)$  to denote arbitrary triple if not otherwise specified. We consider the following formulation.

$$\min_{\substack{\mathbf{B} \in \mathcal{B} \\ \text{supp}(\mathbf{B}) \subseteq K}} \|\mathbf{B}\|_0, \quad (2a)$$

$$\text{s.t. } \mathbf{B} = \arg \min_{\substack{\mathbf{B} \in \mathcal{B} \\ \text{supp}(\mathbf{B}) \subseteq K \\ \theta \in \Theta}} d(f_\theta(\mathbf{B}, c), \mathbf{A}). \quad (2b)$$

The above formulation corresponds to a hierarchical optimization problem [Anandalingam and Friesz, 1992]. The first-level objective, i.e.,  $\|\mathbf{B}\|_0$ , represents the  $l^0$  norm of  $\mathbf{B}$ , and  $\text{supp}(\mathbf{B})$  denotes the support set of the matrix  $\mathbf{B}$ , e.g.,  $\text{supp}(\mathbf{A}) = K$  (definition can be found in Appendix ??). The reconstruction function that takes into account the structure of the graph and the type information of the triples is denoted as  $f_\theta : \mathcal{B} \times c \mapsto \mathcal{B}$ , with a learnable parameter  $\theta$  searched from  $\Theta$ . The reconstruction results are measured by measuring the distance of two matrices, i.e.,

$d : \mathcal{B} \times \mathcal{B} \mapsto \mathbb{R}$ .

To ensure that the compact representation  $\mathbf{B}$  does not degrade into an oversimplified matrix, e.g., a zero matrix,  $\mathbf{0}^{|V| \times |R| \times |V|}$ , rendering it meaningless, and to confirm that the reconstruction process relies on the type information  $c$  rather than overfitting to the structural information  $\mathbf{A}$ , we consider  $f$  to belong to a specific function class. In other words, we want each entry in the function's output, i.e.  $f_\theta(\mathbf{B}, c)[h, r, t]$ , to depend on the type information of  $h, r, t$  and their topological neighbours. This wish is consistent with the idea of message-passing based knowledge graph embedding methods, and here we implement this desired function  $f$  using Relational Graph Convolutional Networks (R-GCN) [Schlichtkrull et al., 2018]. In general, the  $[h, r, t]$ -entry of the output  $f_\theta(\mathbf{B}, c)$  is produced:

$$f_\theta(\mathbf{B}, c)[h, r, t] = s(\mathbf{Z}[h], \mathbf{Z}[r], \mathbf{Z}[t]), \quad (3)$$

where  $s$  denotes any function that maps the embeddings of the triple, i.e.,  $\mathbf{Z}[h], \mathbf{Z}[r], \mathbf{Z}[t] \in \mathbb{R}^d$  into the real range of  $[0, 1]$ . Consequently, the domain of the distance function  $d$  extends to  $[0, 1]^{|V| \times |R| \times |V| \times \mathcal{B}}$ . The embeddings, i.e.,  $\mathbf{Z}[h], \mathbf{Z}[r], \mathbf{Z}[t]$ , are all aggregations of the propagated type information along the graph structure by using multi-layer R-GCN. And the first-layer representations of the entities are initialized as the type information of the entities, i.e.,  $\mathbf{Z}[h]^{(0)} = c[h]$ . After the initialization, each R-GCN layer outputs an updated representation of entities, for example:

$$\mathbf{Z}[h]^{(l+1)} = \sigma \left( \sum_{r \in R} \sum_{j \in \mathcal{N}_h^r} \frac{1}{c_{h,r}} \mathbf{W}_r^{(l+1)} \mathbf{Z}[j]^{(l)} \right) \quad (4)$$

$$+ \mathbf{W}_0^{(l+1)} \mathbf{Z}[h]^{(l)}, \quad (5)$$

where  $\sigma$  is the activation function.  $\mathcal{N}_h^r$  represents the structural neighbor entities of a given entity  $h$  that are connect by a specific relation type  $r$ . Thus, the parameter space of  $f$ , i.e.,  $\Theta$ , consists of all the transformation parameter matrices  $\{\mathbf{W}_r^{(l)}; r \in R, l = 1, \dots, L\}$ , given a hyper-parameter  $L$  representing the depth of the network, the self linear transformation  $\mathbf{W}_0^l$  for each layer, a learnable representation for each relation type, i.e.,  $\mathbf{Z}[r] \in \mathbb{R}^d; r \in R$ , and the possible parameters of  $s$ .

### 3.3 Parameterization and Overall Objective

**Parameterized search.** As presented in Eq. (2a) and (2b), the search space for  $\mathbf{B}$  is discrete and vast, that is,  $(\mathbf{B} \in \mathcal{B}) \wedge (\text{supp}(\mathbf{B}) \subseteq K)$ , making direct optimization of this space challenging. To facilitate computation, we consider parameterizing the search space, transforming the originally discrete space into a continuous one. To achieve this goal, recall that  $\mathbf{B}$  is the most compact representation of  $\mathbf{A}$ . Hence, we naturally

assume that the entry value of  $\mathbf{B}[h, r, t]$ , which indicates whether the corresponding triple  $(h, r, t)$  belongs to the compact representation of the knowledge graph, should be inferred from the neighboring connections and the type information  $c$ . In simple terms, a triple is retained in the compact representation of a knowledge graph if it semantically aligns with the local connections. We implement this by implementing a masking function on the given knowledge graph, i.e.  $m : \mathcal{B} \times c \mapsto \mathcal{B}$ , which determines whether a triple should be retained in the compact representation. We use a design similar to the reconstruction function, as shown below:

$$m_\phi(\mathbf{A}, c)[h, r, t] = s(\mathbf{H}[h], \mathbf{H}[r], \mathbf{H}[t]), \quad (6)$$

where  $\phi$  is the learnable parameter in the masking function, and its parameter space includes all parameters involved in calculating the embeddings denoted by  $\mathbf{H}$  and the possibly potential parameters in  $s$ . The vector  $\mathbf{H}[h] \in \mathbb{R}^d$  is recursively calculated using Eq. (5), initialized in the same way as above. In the implementation of  $s$ , we deploy a Multi-Layer Perceptron (MLP) to enhance the flexibility of inferring the compact set from all triplets. Also note that the parameters of  $f_\theta$  and  $m_\phi$  are independent, as indicated by the different symbols used, i.e.,  $\mathbf{Z}$  and  $\mathbf{H}$  are different, although their internal function classes are identical.

Given this efficient parameterization of the search space, the original optimization problem, as demonstrated in Eqs.(2a) and (2b), is modified into the following form:

$$\min_{\phi \in \Phi} \|m_\phi(\mathbf{A}, c)\|_0, \quad (7a)$$

$$\text{s.t. } m_\phi(\mathbf{A}, c) = \arg \min_{\substack{\phi \in \Phi \\ \theta \in \Theta}} d(f_\theta(m_\phi(\mathbf{A}, c), c), \mathbf{A}). \quad (7b)$$

**Asymptotic discretization.** The parameterization outlined above facilitates continuous optimization using gradient descent. However, it is worth noting that in the modified framework, i.e., Eqs.(7a) and (7b), both the reconstruction function and the masking function produce continuous output, creating a discrepancy between the original problem and its modified version. To address this, we employ suitable techniques to discretize the output.

First, we apply a sigmoid transformation to the output of both functions to polarize the values. Furthermore, we utilize Gumbel-Softmax [Jang et al., 2017] to further discretize the output of the masking function. This is particularly important because the output of the masking function serves as the input to the reconstruction function, which requires stricter discretization. For instance, if  $q_i$  represents an entry of the output of the original masking function, i.e., Eq. (6), the actual input

to the reconstruction function is given by:

$$\frac{\exp(\frac{\log(\frac{1}{1+\exp(-q_i)}+p_i)}{\tau})}{\exp(\frac{\log(\frac{1}{1+\exp(-q_i)}+p_i)}{\tau}) + \exp(\frac{\log(\frac{1}{1+\exp(-(1-q_i))}+p'_i)}{\tau})}, \quad (8)$$

where  $p_i$  and  $p'_i$  are independently sampled from a Gumbel distribution with a location parameter of 0 and a scale parameter of 1, and  $\tau$  is a hyper-parameter controlling the sparsity of the outputs.

**The overall optimization objective and the sparsity constraint.** We consider the following objective to solve the constrained optimization problem formulated in Eqs.7a and 7b.

$$\min_{\substack{\phi \in \Phi \\ \theta \in \Theta}} d(f_\theta(m_\phi(\mathbf{A}, c), c), \mathbf{A}) + \gamma \rho(m_\phi(\mathbf{A}, c)), \quad (9)$$

where  $\rho(m_\phi(\mathbf{A}, c))$  is a sparsity regularizer with a non-zero hyper-parameter  $\gamma$  controlling the strength, for replacing the original  $l^0$  norm constrain, because the original one is the discrete component, making continuous optimization of the whole problem difficult. Considering the relevant discussions on sparsity constraints [Ng et al., 2024], we adopt the minimax concave penalty regularizers [Zhang, 2010], where both the  $l^1$  and  $l^2$  norms are involved:

$$\rho(\mathbf{X}[i, j, k]) = \begin{cases} \lambda |\mathbf{X}[i, j, k]| - \frac{\mathbf{X}[i, j, k]^2}{2\alpha}, & \text{if } |\mathbf{X}[i, j, k]| \leq \alpha\lambda, \\ \frac{\alpha\lambda^2}{2}, & \text{otherwise.} \end{cases}$$

### 3.4 Denoising Based on the Discrepancy

Given the objective formulated by Eq. (9), the optimized functions  $m_\phi$  and  $f_\theta$  can be obtained. Respectively,  $f_{\theta^*}$  is the general mechanism that generates the links by the minimal fundamental set of triples, which is marked by  $m_{\phi^*}(\mathbf{A}, c)$ . Obviously. Our idea is that noisy triples should not be contained in the reconstruction  $f_{\theta^*}(m_{\phi^*}(\mathbf{A}, c), c)$ . Therefore, we can detect noisy triples simply by comparing the discrepancy between the original input  $\mathbf{A}$  and the reconstruction, with a chosen threshold 0.5. More specifically, given a triple  $(h, r, t)$  that appears in the original knowledge graph, that is,  $\mathbf{A}[h, r, t] = 1$ , the noise label  $y((h, r, t))$  is determined in the following form:

$$y((h, r, t)) = \begin{cases} 1, & \text{if } f_{\theta^*}(m_{\phi^*}(\mathbf{A}, c))[h, r, t] \geq 0.5 \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, the task of completing the knowledge graph can be accomplished by introducing the triples that have entry values greater than or equal to the threshold. And the compression task is already done by obtaining the fundamental representation of the knowledge graph,

i.e.,  $m_{\phi^*}(\mathbf{A}, c)$ . The sensitivity analysis of choosing the threshold is provided in the Appendix. In addition, the diagram demonstration of the training and inference phases can be found in Figure 5(a) and Figure 5(b).

## 4 DISCUSSIONS

**The analogy to single image denoising.** To make it more intuitive, let us draw the analogy between our knowledge graph denoising framework and single image denoising techniques [Quan et al., 2020, Huang et al., 2021, Ko and Lee, 2023]. In these methods, a self-supervised approach is employed, where no reference image is provided, and thereby, the original image is used as input to the denoiser, which outputs an image of the same size. The assumption is that local patches of the image, regardless of their location, should preserve the same noise pattern. This allows the denoiser to be trained on multiple local patches. Similarly, our knowledge graph denoising proposal assumes a shared noise pattern at the type information level, which abstracts the token-level information we observe. In this way, “local patches” can be constructed using the different token-level embodiments of the abstract type information. We would like to add that the proposed framework is expected to help in knowledge graph compression and completion as well, although we do not focus on them [Zhang et al., 2020, Sachan, 2020, Zamini et al., 2022].

**Sample size and the type abstraction.** Just as in image denoising, where a larger image provides more local patches to train a more stable and generalizable denoising model (while avoiding converging to identity mapping), knowledge graph denoising also benefits from a sufficiently large knowledge graph. Thus, our experiments focus on real-world knowledge graphs to ensure a sufficient scale for effective denoising. It is crucial that the abstraction of the knowledge graph should not be too high for an effective denoising, as overly abstract type information can make it difficult for the model to fit the input graph information. Finally, we would like to note that the compact representation proposed in our approach is essential for denoising tasks, since it reduces the chance that the model performs identity mapping by storing the most informative part of the graph, ensuring that noise is not preserved.

## 5 EXPERIMENTS

To evaluate our proposal for detecting noise, we focus on uncovering noise directly from the original dataset, given that the actual distributions of noise in real-world knowledge graphs are typically unknown. This approach, which aims to reveal noise in real-world

datasets, has been relatively unexplored in existing research. Our experiments focus on two sets of questions. **RQ1:** Is the identified noise reasonable? Can other knowledge graph embedding methods achieve similar noise detection? **RQ2:** Does the proposed method avoid fitting noisy data? How do the individual components of our proposed method contribute to this attribute? **RQ3:** Is our proposal robust to mild corruption of type information? Due to page limit, we save i) noise detection verification of general setting (i.e., uncover the additionally added abnormal triples from the original knowledge graph) and the results on larger-scale datasets, ii) hyper-parameter sensitivity analysis, iii) computational complexity examination, and iv) preliminary evaluation of our proposal on knowledge graph completion and compression task in the Appendix.

## 5.1 Settings

**Dataset and baselines.** We conduct experiments mainly on three datasets: WN18RR [Bordes et al., 2013], FB15k-237 [Toutanova and Chen, 2015], and NELL-995 [Xiong et al., 2017]. In addition, the larger-scale datasets are also considered, namely, ogbl-biokg [Hu et al., 2020], DBpedia [Lehmann et al., 2015], and Yago [Mahdisoltani et al., 2013], which are put in the Appendix due to page limit. The entity type information for the WN18RR and FB15k-237 datasets was obtained from the publicly available OpenKE project<sup>1</sup>, where the entity type information is extracted based on the relation type. For FB15k-237, we obtained the entity types from the corresponding text descriptions.

For the baselines, here we choose embedding-based denoising approaches for fair comparisons and also due to the limit of page. For other types of baseline, please refer to the Appendix. Among them, the most related are the MPNN-based embedding approaches: R-GCN [Schlichtkrull et al., 2018], KBGAT [Nathani et al., 2019], and the denoising methods GCNN [Jia et al., 2019a], and TRUST [Neil et al., 2018]. We also compare MLP-enhanced distance measurement methods, namely DistMult [Yang et al., 2015] and ConvE [Dettmers et al., 2018]. Their superiority compared to vanilla representation space modeling methods has been revealed [Li et al., 2023]; therefore, we do not additionally include those methods as baselines. In these methods, negative sampling is usually employed to enhance robustness. Because we are concerned with revealing noise in real-world datasets, rather than revealing additional noise added

Method	NELL-995	WN18RR	FB15k-237
DistMult	1.50 ± 1.58	0.50 ± 0.85	3.20 ± 1.23
ConvE	1.90 ± 0.24	1.22 ± 0.24	4.60 ± 3.24
KBGAT	0.50 ± 0.76	2.83 ± 1.72	15.00 ± 22.49
R-GCN	1184.50 ± 675.65	442.00 ± 124.45	343.00 ± 87.88
GCNN	0.65 ± 0.27	2.25 ± 1.90	12.23 ± 3.44
TRUST	1.84 ± 0.73	3.15 ± 1.55	13.51 ± 15.43
RAE (Ours)	25.00 ± 1.00	49.25 ± 42.17	120.67 ± 5.03

Table 2: Comparisons of the number of detected noise among different approaches (#E).

to the original dataset, these suggestions for combating robustness are already embodied in the knowledge graph embedding methods described above.

**Implementations.** Like most MPNN-based knowledge graph embedding approaches that do not rely on strict logical rules, we typically process triples by appending a reverse version of each. This practice fully leverages the asymmetric dependencies between triples. For example, the example we give in Section 1, (`concept_personasia_news_corporation`, `concept:agentcontrols_reverse`, `concept_city_murdoch`) is also included in the training set of these approaches. In our implementation, we adopt this practice, considering that type dependencies are also asymmetric. This modification further strengthens the type dependencies our method relies on. For the hyper-parameters of the masking and reconstruction functions, we use the optimized settings suggested by recent work [Li et al., 2023], and we provide detailed settings in Appendix A.1.1. Since our proposed method is an auto-encoder architecture based on R-GCN, the abbreviation RAE (i.e., R-GCN Auto-Encoder) denotes our proposal. More implementation details regarding RAE and baselines are provided in the Appendix <sup>2</sup>.

**Evaluation metrics.** We first focus on revealing the noise triples from real-world data, where the true error label is unknown. In this case, we report the number of uncovered noisy triples for **RQ1** and **RQ3**, which is computed as:  $\#E = |\{f_{\theta^*}(m_{\phi^*}(\mathbf{A}, c)) [h, r, t] \geq 0.5 : \mathbf{A}[h, r, t] = 1\}|$ . In the Appendix, we elaborate the metrics we use for evaluating the performance of artificially added error triples (i.e., RQ i)), when all the observed triples are assumed to be correct and the noise labels are known, including true negative rate given a bandit.

## 5.2 Experimental Results on Real-World Data

**RQ1: noise detection and case study on NELL-995.** We employ the reconstruction discrepancy-based

<sup>1</sup><https://github.com/thunlp/OpenKE>

<sup>2</sup>The link to our code repository is <https://github.com/sajqavril/Code-Repo-for-R-GCN-Auto-Encoder.git>.

Head Entity	Relation	Tail Entity
person_larry_page	agentcontrol	university_google
stateorprovince_idaho	stateorprovinceisborderedbystateorprovince	stateorprovince_south_dakota
personasia_news_corporation	agentcontrol	city_murdoch
Eight non-noisy triples have been omitted due to page limitations.		
ceo_robert_iger	agentcontrol	person_disney
company_apple002	headquartered	city_london_city
personmexico_m_s	agentcontrol	director_stuart_rose
Four non-noisy triples have been omitted due to page limitations.		
profession_parts	proxyfor	book_new
sportsleague_irl	agentcontrol	personmexico_tony_george
Four non-noisy triples have been omitted due to page limitations.		
movie_repo	synonymfor	airport_epel
recordlabel_roc_a_fella	organizationterminatedperson	ceo_damon_dash

Table 3: Detected noise from the NELL-995 dataset (best viewed in color).

Corrupt. ratio	NELL-995	WN18RR	FB15k-237
w/o	25.00 ± 1.00	49.25 ± 42.17	120.67 ± 5.03
w/ 0.01%	23.37 ± 1.37	45.73 ± 51.32	118.94 ± 3.11
w/ 0.1%	26.09 ± 2.02	44.25 ± 52.87	124.85 ± 6.36
w/ 1.0%	21.57 ± 4.29	52.36 ± 58.66	110.61 ± 10.30

Table 4: Robustness w.r.t. corrupted type labels ( $\#E$ ).

noise detection method outlined in Section 3 to identify potential noise in three real-world datasets. Table 2 shows that our proposed method consistently detects a stable number of noisy triples with lower variance, except for WN18RR. A possible reason is that this dataset is characterized by overly abstract type information (as shown in Table 1), which poses challenges to our proposal to effectively fit the triples. In contrast, the baselines either vary widely in the number of detected noisy triples, such as R-GCN, or converge on nearly all the triples, leaving little room for denoising, such as DistMult, KBGAT, GCNN, and TRUST. We also include the DistMult training process in Appendix A.3 to confirm its fit to all triples. Table 3 shows the noise detected from NELL-995 by our proposal. For readability, we remove the “concept” prefix. Obvious noise is highlighted (some non-noise triples are omitted; please refer to Appendix A.4 for the complete list). In particular, most noise triples are detected by their reverse counterparts. For example, the noisy triple (ceo\_robert\_iger, agentcontrol, person\_disney) is originally captured by its reverse counterpart (person\_disney, agentcontrol\_reverse, ceo\_robert\_iger). This finding further supports our approach of reversing links to better utilize the type constraints contained in the triples.

**RQ2: avoiding fitting on noise.** As we analyze in Section 3, the triples are distributed centralized with respect to the types. Generally speaking, if a triple has a type that agrees with the majority of the triples’ type, then it is less likely to be a noisy triple. Given that the ground truth of noisy triples is unknown in real-world

datasets, we indirectly demonstrate how the model avoids fitting noise by showing how well it fits the more frequently occurring types of triple. In Figure 2, we compare the performance of R-GCN, our proposed RAE, RAE without type information guidance (RAE w/o T), and RAE without Gumbel-Softmax (RAE w/o GS). RAE w/o T treats each entity as a different token like R-GCN, maintaining the auto-encoder framework. RAE w/o GS removes the Gumbel-Softmax component. Each point in the figure represents a triple type, and the y-axis score for each type is the average score of all triples of that type minus the scores of 10 randomly sampled negative triples (created by replacing the tail entities). For example, given a triple type  $(c_h, r, c_t)$ , the score is calculated as:

$$\#F = \frac{1}{|S|} \sum_{(h,r,t) \in S} (\hat{\mathbf{A}}[h, r, t] - \frac{1}{10} \sum_{(h', r', t') \in S'} \hat{\mathbf{A}}[h', r', t']), \quad (10)$$

where  $S = \{(h, r, t) : c[h] = c_h, c[r] = r, c[t] = c_t\}$  marks the set of all the triples that have the type of  $(c_h, r, c_t)$ , and  $\hat{\mathbf{A}} = f_{\theta^*}(m_{\phi^*}(\mathbf{A}, c), c)$  is the reconstruction.  $(h', r', t')$  is the corresponding corrupted negative triple, each of which is from  $S'$ , whose entity or relation is randomly corrupted. From the results, we find that only the complete RAE model effectively captures high-frequency type triples. The comparison between R-GCN and RAE w/o GS shows that the asymptotic discretization achieved through the Gumbel-Softmax contributes to better capturing the type constraints among the triples. The procedure of drawing this picture is put in the Appendix.

### RQ3: robustness for corrupted type information

Since we explicitly reveal the type-inconsistent noise in our method, the robustness of our proposal with respect to type corruption is worth investigating. Consequently, in Table 4 we compare the performance ( $\#E$ ) when different fractions of the entitytype labels are corrupted, from 0 to 1.0%, where we find that the proposal is quite robust to the acceptable fraction of corruption.



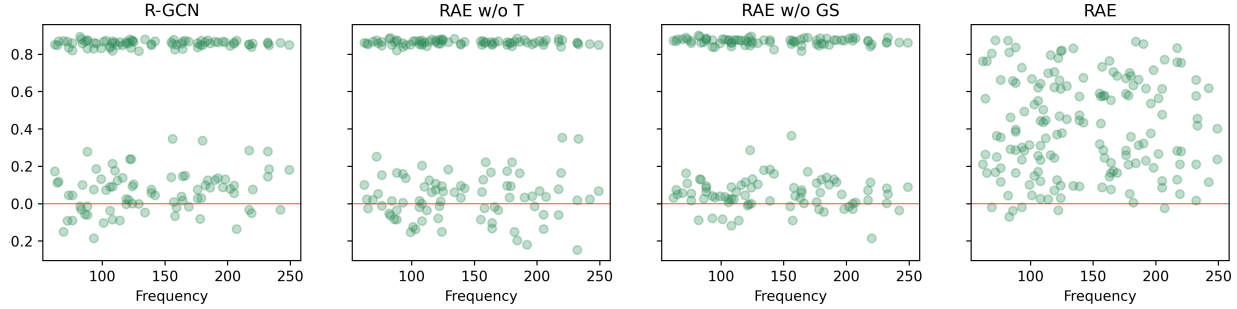


Figure 2: Comparisons of different methods on fitting triples with various type frequencies ( $\#F$ ).

## 6 RELATED WORK

**Knowledge graph denoising.** There are mainly three types of methods to deal with noise in knowledge graphs. The first is the use of logical rules, which considers the noisy triples as those that violate the predefined rules [Galárraga et al., 2013, Pellissier Tanon et al., 2017, Cheng et al., 2018]. However, these methods make too strict assumptions about the correct triples of following the logical reasoning rules, which may not be compatible with real-world knowledge graph constructions. Except for such hard rule constraints, some work exploits soft rules, that is, inductive combinations of the rules defined by the type or attributes information of the triples [Belth et al., 2020]. In both cases, it is still possible that there are unknown additional constraints between edges other than explicit rules, so it is desirable to develop an automatic approach to discover all such constraints from the data. Recently, a framework for denoising common-sense knowledge graphs based on pre-train language models [Deng et al., 2023] has emerged, aiming to use external knowledge to denoise common-sense knowledge graphs. For such methods to work, except for the assumption that the additional supervision input is correct, it requires a high quality of entity alignment for the external knowledge to help, and the cost of obtaining such knowledge is not trivial. The third class of methods uses knowledge graph embedding techniques to give rise to a confidence score (sometimes called trustworthiness) of the triples, which is generated based on the structure of the knowledge graph [Paulheim and Gangemi, 2015, Neil et al., 2018, Jia et al., 2019b, Xu et al., 2022, Zhang et al., 2022, Zhang et al., 2023]. These works either rely on topological information without semantic support, which may lead to overfitting on the structure if a given knowledge graph (e.g., see the results shown in Table 2). Or, expensive attributes of the entities are required [Zhang et al., 2023], which is not feasible for most real-world scenarios.

**Remark.** From the related work discussed, except

for those that require external knowledge, we can see that many of them are trying to reveal noise by imposing regularities among the triples, by logical rules [Cheng et al., 2018, Belth et al., 2020], or by extracting the invariant among the changes [Dong et al., 2023, Zhang et al., 2022]. In addition, the importance of semantic information beyond the topological relationships captured by embedding-based approaches [Zhao et al., 2019] has been pointed out [Zhang et al., 2023, Belth et al., 2020]. Therefore, our proposal integrates these merits and uses type information to reveal the noisy triples by enforcing type-consistent constraints on knowledge graphs. Finally, our explicit assumptions about the noise we are trying to tackle allow us to evaluate directly on real knowledge graphs without error labels, which is harder than revealing random false positive triples.

## 7 CONCLUSIONS

This work investigates the feasibility of incorporating type information to identify noise in knowledge graphs. Given the definition of type-inconsistency noise, we propose a self-supervised approach based on type information to denoise knowledge graphs. This approach leverages the readily available type information in datasets and demonstrates its ability to avoid fitting noisy triples and effectively detect potential noise in real-world datasets.

**Limitations and future work.** Our proposal still faces challenges in distinguishing isolated facts, such as triples that are independent of existing triples, and noise of type inconsistency, which is consistent with our analysis. More research is needed to differentiate these facts, possibly by analyzing the distributional differences between the two sets. In scenarios where the embedded type information in the dataset is limited—either in quantity or quality—we plan to explore multi-hop combinations of relations and entity types to enhance the support from type information. In addition, the interpretability of the masking and reconstruction functions needs to be improved.



## 8 ACKNOWLEDGEMENTS

All the authors would like to thank the anonymous reviewers for their generous feedback. We would also like to acknowledge the support from NSF Award No. 2229881, AI Institute for Societal Decision Making (AI-SDM), the National Institutes of Health (NIH) under Contract R01HL159805, and grants from Quris AI, Florin Court Capital, and MBZUAI-WIS Joint Program.

## References

- [Anandalingam and Friesz, 1992] Anandalingam, G. and Friesz, T. L. (1992). Hierarchical optimization: An introduction. *Annals of Operations Research*, 34:1–11.
- [Belth et al., 2020] Belth, C., Zheng, X., Vreeken, J., and Koutra, D. (2020). What is normal, what is strange, and what is missing in a knowledge graph: Unified characterization via inductive summarization. In *Proceedings of The Web Conference 2020*, pages 1115–1126.
- [Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.
- [Cheng et al., 2018] Cheng, Y., Chen, L., Yuan, Y., and Wang, G. (2018). Rule-based graph repairing: Semantic and efficient repairing methods. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 773–784. IEEE.
- [Chetoui et al., 2022] Chetoui, I., El Bachari, E., El Adnani, M., and El Hassan, A. (2022). Graph neural networks to improve knowledge graph embedding: A survey. In *International Conference on Advanced Intelligent Systems for Sustainable Development*, pages 15–25. Springer.
- [Deng et al., 2023] Deng, Z., Wang, W., Wang, Z., Liu, X., and Song, Y. (2023). Gold: A global and local-aware denoising framework for commonsense knowledge graph noise detection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3591–3608.
- [Dettmers et al., 2018] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.
- [Dong et al., 2023] Dong, J., Zhang, Q., Huang, X., Tan, Q., Zha, D., and Zihao, Z. (2023). Active ensemble learning for knowledge graph error detection. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 877–885.
- [Galárraga et al., 2013] Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. (2013). Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422.
- [Guo et al., 2020] Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., and He, Q. (2020). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.
- [Hu et al., 2020] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.
- [Huang et al., 2021] Huang, T., Li, S., Jia, X., Lu, H., and Liu, J. (2021). Neighbor2neighbor: Self-supervised denoising from single noisy images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14781–14790.
- [Jang et al., 2017] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.
- [Ji et al., 2021] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.
- [Jia et al., 2019a] Jia, S., Xiang, Y., Chen, X., and Wang, K. (2019a). Triple trustworthiness measurement for knowledge graph. In *The World Wide Web Conference*, pages 2865–2871.
- [Jia et al., 2019b] Jia, S., Xiang, Y., Chen, X., and Wang, K. (2019b). Triple trustworthiness measurement for knowledge graph. In *The World Wide Web Conference*, pages 2865–2871.
- [Ko and Lee, 2023] Ko, J. and Lee, S. (2023). Self2self+: Single-image denoising with self-supervised learning and image quality assessment loss. *arXiv preprint arXiv:2307.10695*.

- [Lehmann et al., 2015] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.
- [Li et al., 2023] Li, J., Shomer, H., Ding, J., Wang, Y., Ma, Y., Shah, N., Tang, J., and Yin, D. (2023). Are message passing neural networks really helpful for knowledge graph completion? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10696–10711.
- [Ma et al., 2023] Ma, T., Chen, Y., Tao, W., Zheng, D., Lin, X., Pang, P. C.-l., Liu, Y., Wang, Y., Song, B., and Zeng, X. (2023). Learning to denoise unreliable interactions for link prediction on biomedical knowledge graph. *arXiv preprint arXiv:2312.06682*.
- [Mahdisoltani et al., 2013] Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2013). Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.
- [Nathani et al., 2019] Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. (2019). Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723.
- [Neil et al., 2018] Neil, D., Briody, J., Lacoste, A., Sim, A., Creed, P., and Saffari, A. (2018). Interpretable graph convolutional neural networks for inference on noisy knowledge graphs. *arXiv preprint arXiv:1812.00279*.
- [Ng et al., 2024] Ng, I., Zheng, Y., Dong, X., and Zhang, K. (2024). On the identifiability of sparse ica without assuming non-gaussianity. *Advances in Neural Information Processing Systems*, 36.
- [Pan et al., 2024] Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., and Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.
- [Paulheim and Gangemi, 2015] Paulheim, H. and Gangemi, A. (2015). Serving dbpedia with dolce—more than just adding a cherry on top. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I 14*, pages 180–196. Springer.
- [Pellissier Tanon et al., 2017] Pellissier Tanon, T., Stepanova, D., Razniewski, S., Mirza, P., and Weikum, G. (2017). Completeness-aware rule learning from knowledge graphs. In *The Semantic Web-ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*, pages 507–525. Springer.
- [Quan et al., 2020] Quan, Y., Chen, M., Pang, T., and Ji, H. (2020). Self2self with dropout: Learning self-supervised denoising from single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1890–1898.
- [Sachan, 2020] Sachan, M. (2020). Knowledge graph embedding compression. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2681–2691.
- [Schlichtkrull et al., 2018] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.
- [Toutanova and Chen, 2015] Toutanova, K. and Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.
- [Xiong et al., 2017] Xiong, W., Hoang, T., and Wang, W. Y. (2017). Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.
- [Xu et al., 2022] Xu, Z., Huang, X., Zhao, Y., Dong, Y., and Li, J. (2022). Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 444–457. Springer.
- [Yang et al., 2015] Yang, B., Yih, S. W.-t., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.
- [Zamini et al., 2022] Zamini, M., Reza, H., and Rabiei, M. (2022). A review of knowledge graph completion. *Information*, 13(8):396.
- [Zeng et al., 2021] Zeng, K., Li, C., Hou, L., Li, J., and Feng, L. (2021). A comprehensive survey of entity alignment for knowledge graphs. *AI Open*, 2:1–13.
- [Zhang, 2010] Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, pages 894–942.

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

[Zhang et al., 2023] Zhang, Q., Dong, J., Tan, Q., and Huang, X. (2023). Integrating entity attributes for error-aware knowledge graph embedding. *IEEE Transactions on Knowledge and Data Engineering*.

[Zhang et al., 2020] Zhang, S., Zhang, Z., Zhuang, F., Shi, Z., and Han, X. (2020). Compressing knowledge graph embedding with relational graph auto-encoder. In *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 366–370. IEEE.

[Zhao et al., 2020] Zhao, X., Zeng, W., Tang, J., Wang, W., and Suchanek, F. M. (2020). An experimental study of state-of-the-art entity alignment approaches. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2610–2625.

[Zhao et al., 2019] Zhao, Y., Feng, H., and Gallinari, P. (2019). Embedding learning with triple trustiness on noisy knowledge graph. *Entropy*, 21(11):1083.

## Checklist

1. For all models and algorithms presented, check if you include:
  - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
  - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
  - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
  - (a) Statements of the full set of assumptions of all theoretical results. [Yes]
  - (b) Complete proofs of all theoretical results. [Yes]
  - (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
  - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
  - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
  - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
  - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
  - (a) Citations of the creator If your work uses existing assets. [Yes]
  - (b) The license information of the assets, if applicable. [Yes]
  - (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
  - (d) Information about consent from data providers/curators. [Yes]
  - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
  - (a) The full text of instructions given to participants and screenshots. [Not Applicable]
  - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
  - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

## Contents (Appendix)

<b>A MORE EXPERIMENTS ON DENOISING TASK</b>	<b>12</b>
A.1 Experimental Settings . . . . .	12
A.1.1 Hyper-Parameters Settings . . . . .	12
A.1.2 Computation Architecture and Optimization Methods . . . . .	13
A.2 Comprehensive Visualization of the Type Distribution . . . . .	13
A.3 Converging Process of Compared Methods . . . . .	14
A.4 Full Set of Detected Noisy Triples from NELL-995 . . . . .	15
A.5 Parameter Sensitivity Analysis and Robustness Study . . . . .	15
<b>B DIAGRAM DEMONSTRATION OF THE PROPOSED RAE</b>	<b>16</b>
<b>C BOARDER IMPACTS</b>	<b>16</b>

## A MORE EXPERIMENTS ON DENOISING TASK

### A.1 Experimental Settings

#### A.1.1 Hyper-Parameters Settings

In the experiments, we refer to the framework proposed by the analytical work between MPNN-based and MLP-based knowledge graph embedding framework [Li et al., 2023]. The hyper-parameters of the models are listed as below:

- R-GCN:
  - FB15k-237: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1.
  - NELL-995: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1.
  - WN18RR: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1.
- RAE (Ours):
  - FB15k-237: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1, sparsity\_constrain=0.5.
  - NELL-995: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1, sparsity\_constrain=0.5.
  - WN18RR: negative\_sampling=10, score\_function=DistMult, num\_blocks=100, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, hidden\_dropout=0.1, sparsity\_constrain=0.5.
- KBGAT:
  - FB15k-237: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, num\_heads=2, hidden\_dropout=0.3.
  - NELL-995: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, num\_heads=8, hidden\_dropout=0.3.
  - WN18RR: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, gc\_n\_layer=2, num\_heads=8, hidden\_dropout=0.3.
- MLP-DistMult:
  - FB15k-237: negative\_sampling=0, score\_function=DistMult, learning\_rate=0.0001, batch=512, l2=0, num\_workers=3, layer=2, hidden\_dropout=0.05.
  - NELL-995: negative\_sampling=0, score\_function=DistMult, learning\_rate=0.0001, batch=512, l2=0, num\_workers=3, layer=2, hidden\_dropout=0.05.

- WN18RR: negative\_sampling=0, score\_function=DistMult, learning\_rate=0.0001, batch=512, l2=0, num\_workers=3, layer=2, hidden\_dropout=0.05.

- MLP-ConvE:

- FB15k-237: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, layer=1, hidden\_dropout=0.3.
- NELL-995: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, layer=1, hidden\_dropout=0.3.
- WN18RR: negative\_sampling=0, score\_function=ConvE, learning\_rate=0.001, batch=512, l2=0, num\_workers=3, layer=1, hidden\_dropout=0.3.

In addition, for our method, the hyper-parameters  $\alpha$  and  $\lambda$  in the minimax concave regularizer are set to 10 and 1, respectively. For the strength of the sparsity constrain, we conducted comparisons between  $\{0.1, 0.5, 1.0\}$  without observing significant differences, and we chose 0.5 as the default setting when no specific note is provided.

In addition, for all the reported numbers, we ran the experiments five times independently with random seeds chosen from  $\{41504, 42, 0, 1, 2\}$  and calculated the mean and variance values. Exceptions were made for the case study, as it is challenging to calculate the average and variance across different runs. However, we additionally provide the results from other runs, see the Tables below, where we found the differences between different runs were not significant, as demonstrated in Table 2.

### A.1.2 Computation Architecture and Optimization Methods

On the hardware side, all experiments were conducted on a machine equipped with a 24 vCPU Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz and 2 GPUs (RTX 4090, each with 24GB of memory). On the software side, CUDA 12.1 was used, and the computational platform primarily consisted of PyTorch 2.1.0 and Python 3.10 running on Ubuntu 22.04. All optimization problems were solved using the Adam optimizer with a default weight decay of 0.00005. The learning rate varied across different baselines and datasets, and detailed learning rate settings are provided in the appended hyper-parameters settings section A.1.1.

## A.2 Comprehensive Visualization of the Type Distribution

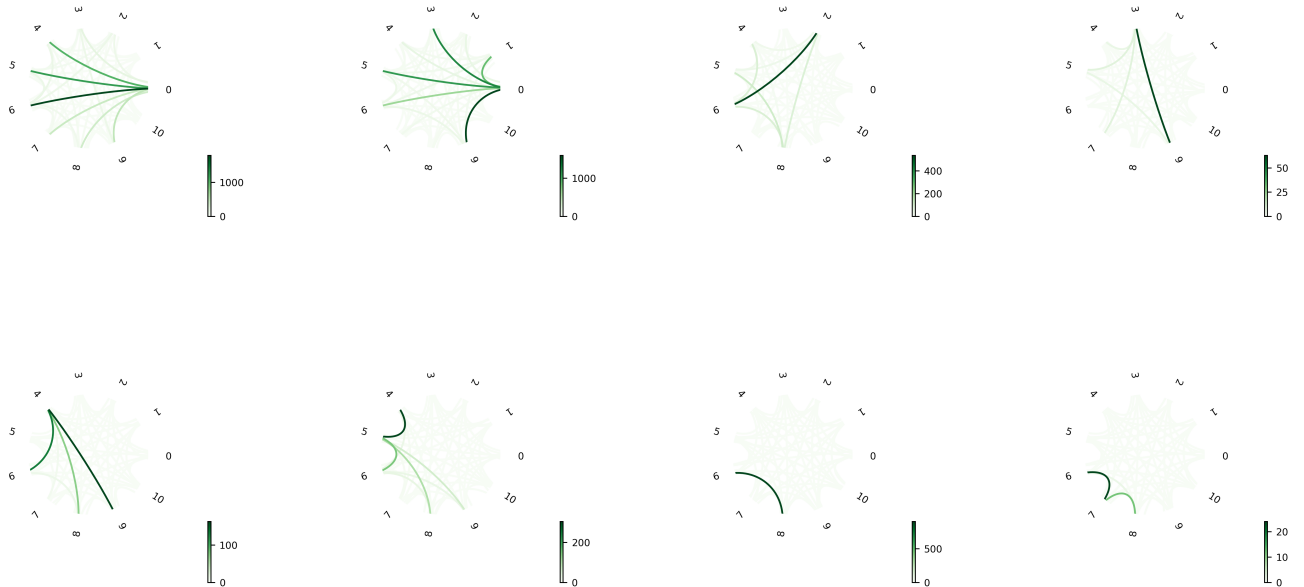


Figure 3: Eight relation type distributions on WN18RR dataset.

Except for Figure 1, which we showed to motivate our proposal in the main content, Figure A.2 shows more relation type distributions corresponding to the entity types, where we can see the legitimate relation types are tightly distributed across all possible combinations of entity types.

### A.3 Converging Process of Compared Methods

As shown in Figure 4(a), Figure 4(b) and Figure 4(c), the convergence processes of the difference methods are compared when applied to the three data sets, i.e. WN18RR, NELL-995 and FB15k-237. From the results we can observe that both DistMult and KBGAT suffer from an overfitting problem, as the number of unfitted triples tends to zero, especially DistMult. This confirms our previous claim that embedding-based methods tend to fit all topological information and cannot effectively use higher-level semantic information, e.g. type information, to detect noise.

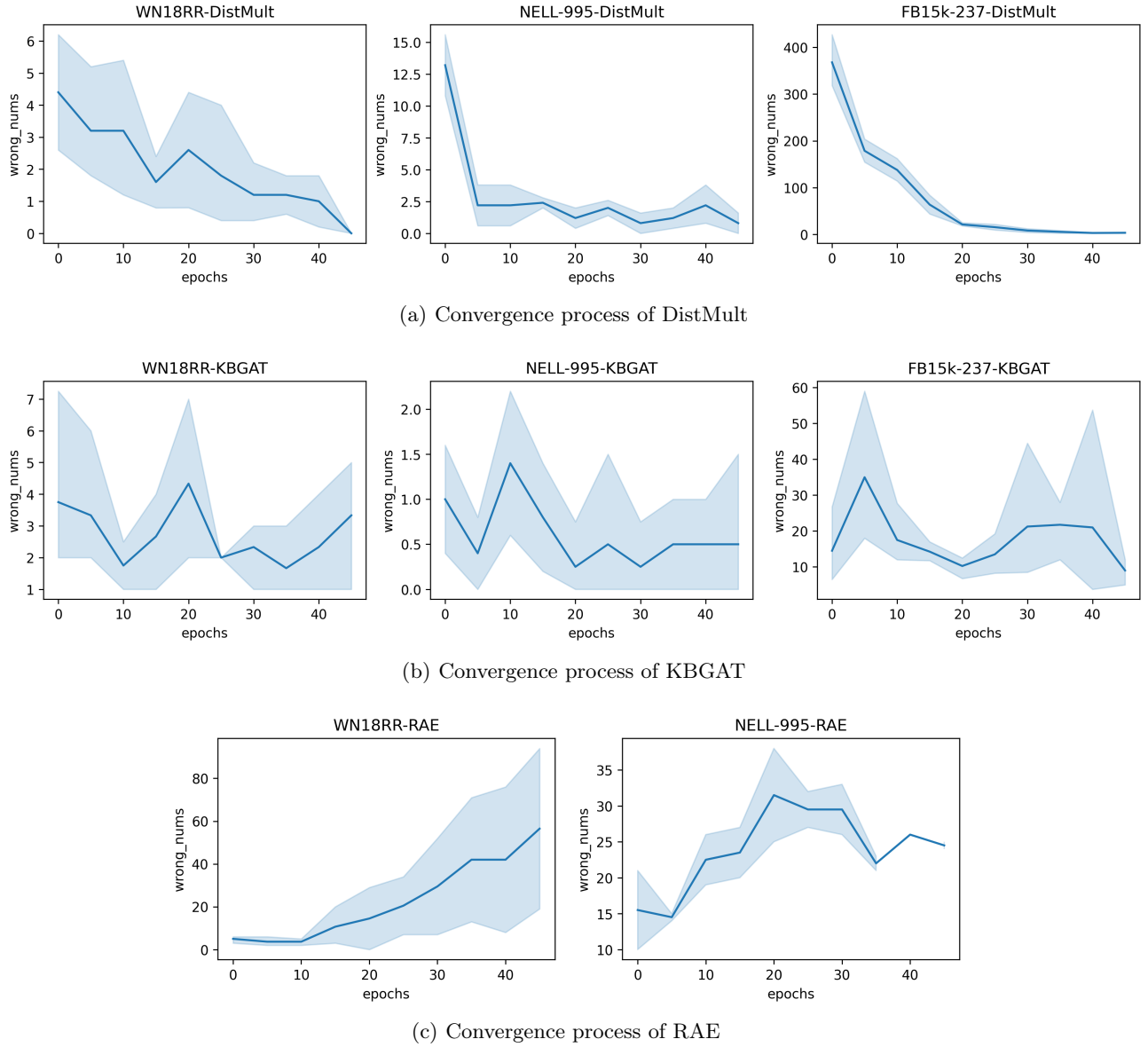


Figure 4: Comparisons of convergence processes of different models



Head Entity	Relation	Tail Entity
person_larry_page	agentcontrol	university_google
stateorprovince_idaho	stateorprovinceisborderedbystateorprovince	stateorprovince_south_dakota
personasia_news_corporation	agentcontrol	city_murdoch
biotechcompany_delta_air_lines_inc	organizationterminatedperson	journalist_richard_anderson
magazine_gucci	organizationterminatedperson	ceo_domenico_de_sole
stateorprovince_idaho	stateorprovinceisborderedbystateorprovince	visualizablescene_washington
stateorprovince_idaho	stateorprovinceisborderedbystateorprovince	stateorprovince_oregon
sportsteam_detroit_tigers	teamplaysport	sport_baseball
person_wendy001	persongraduatedfromuniversit	university_state_university
musicartist_the_jam	agentcollaborateswithagent	male_bruce_foxton
bank_bear_stearns	organizationterminatedperson	ceo_james_cayne
ceo_robert_iger	agentcontrol	person_disney
company_apple002	headquarteredin	city_london_city
personmexico_m_s	agentcontrol	director_stuart_rose
school_shichahai_school	agentcontrol	island_zhang
website_technorati	competeswith	blog_google
sportsteam_arizona_state_sun_devils	agentcollaborateswithagent	personmexico_ncaa
city_erie	proxyfor	stateorprovince_pennsylvania
profession_parts	proxyfor	book_new
sportsleague_irl	agentcontrol	personmexico_tony_george
professor_richard_stallman	personleadsorganization	nonprofitorganization_free_software_foundation
website_cnn_fox	competeswith	newspaper_times
ceo_william_r_klesse	agentcontrol	petroleumrefiningcompany_valero_energy
company_pinco	agentcontrol	wine_gross
website_youtube	competeswith	website_yahoo
movie_repo	synonymfor	airport_epel
recordlabel_roc_a_fella	organizationterminatedperson	ceo_damon_dash

Table 5: Full set of detected noise from NELL-995 dataset (better view in color).

#### A.4 Full Set of Detected Noisy Triples from NELL-995

Table 5 shows all the detected noisy triples.

#### A.5 Parameter Sensitivity Analysis and Robustness Study

We additionally study the sensitivity of our proposal, with respect to the depth of the R-GCN, and the strength of the sparsity constraint. From Table 6 we can see that the different values (e.g. the choice of 0.1, 0.5 and 1.0) of  $\gamma$  do not affect the noise detection performance of the RAE in different data sets. Table 7 shows how the depth of the R-GCN model used in both the encoder and the decoder can affect the noise detection performance of the RAE. From the results we can see that as long as the expressiveness of the model becomes sufficient, e.g. two layers for the datasets used in the experiments, the noise detection remains quite stable thereafter.

$L$	NELL-995	WN18RR	FB15k-237
1	$44.92 \pm 16.06$	$113.48 \pm 32.90$	$90.65 \pm 15.62$
2	$25.00 \pm 1.00$	$49.25 \pm 42.17$	$120.67 \pm 5.03$
3	$22.70 \pm 1.43$	$58.21 \pm 31.69$	$114.55 \pm 4.58$
4	$23.80 \pm 0.61$	$41.77 \pm 29.09$	$126.54 \pm 3.90$

Table 6: Sensitivity study on the depth of R-GCN  $L$

$\gamma$	NELL-995	WN18RR	FB15k-237
0.1	$23.32 \pm 1.57$	$43.77 \pm 22.91$	$131.44 \pm 4.81$
0.5	$25.00 \pm 1.00$	$49.25 \pm 42.17$	$120.67 \pm 5.03$
1.0	$21.56 \pm 2.03$	$58.74 \pm 36.22$	$122.62 \pm 7.40$

Table 7: Sensitivity study on the sparsity constraint  $\gamma$

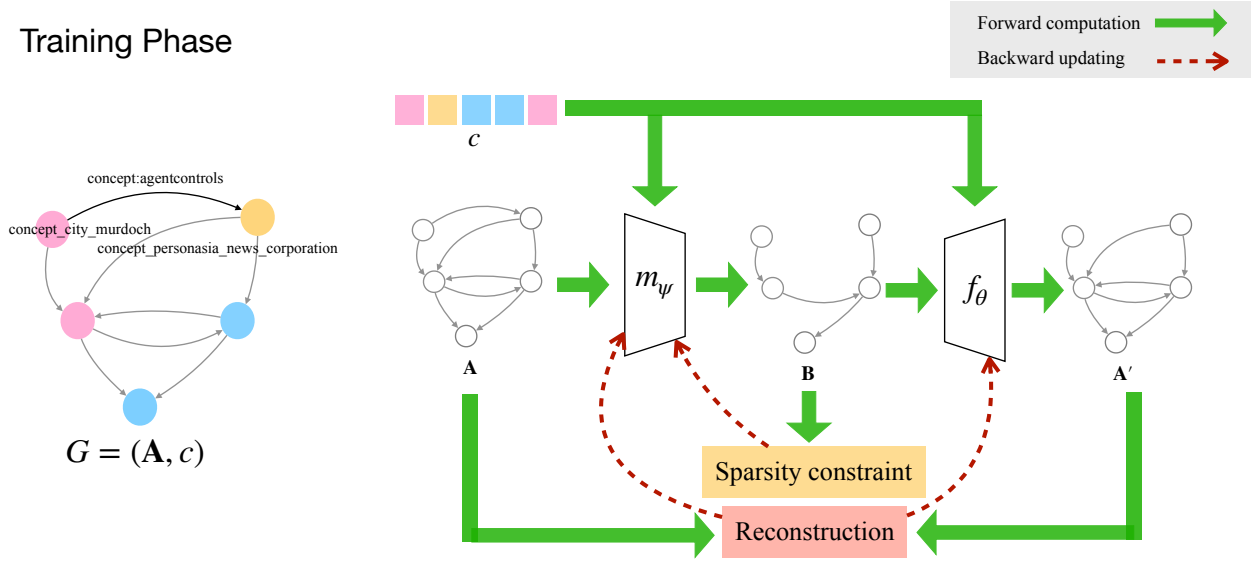
## B DIAGRAM DEMONSTRATION OF THE PROPOSED RAE

The training and inference phases are demonstrated in Figure 5(a) and Figure 5(b). Due to some adjustments in the structure of the appendix, the Figure 7 and Figure 8 that we put in the main content are actually Figure 4(a) and Figure (b) here.

## C BOARDER IMPACTS

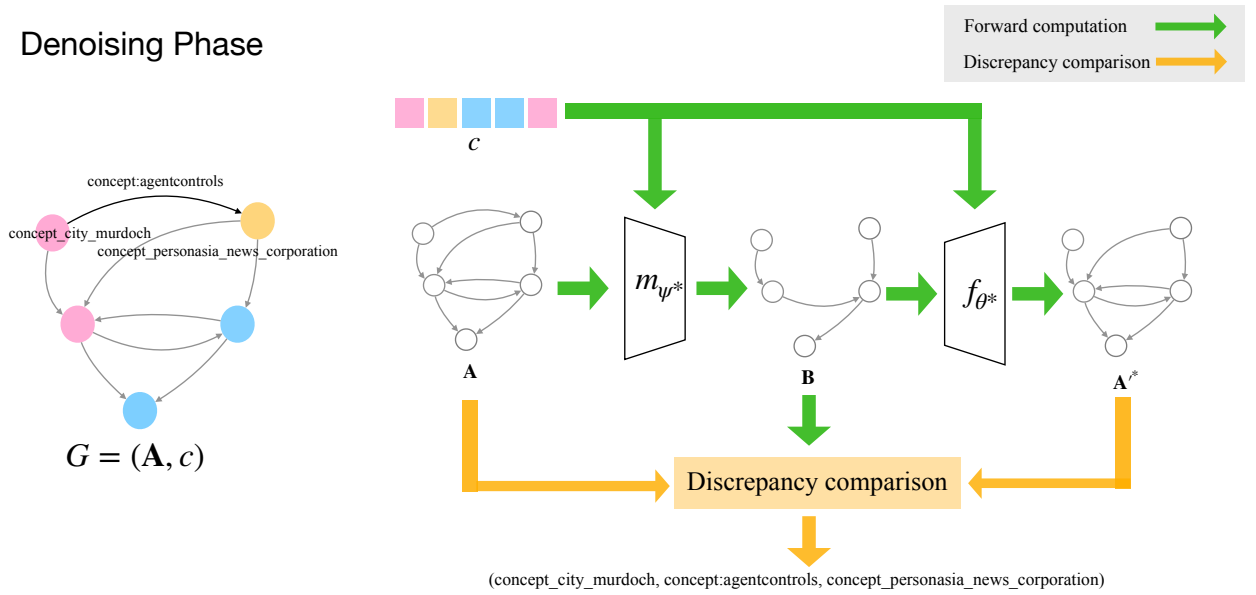
This work has the potential to enhance the reliability of AI applications that incorporate knowledge graphs as external expert knowledge support. However, since the interpretability of the model is not confirmed, it might be used to deliberately reject some facts by a third party. Therefore, the usage of this method should ensure the integrity of the original dataset, preventing any malicious manipulations.

## Training Phase



(a) Training Phase of RAE

## Denoising Phase



(b) Denoising Phase of RAE

Figure 5: Diagram demonstration of the training and inference phases of our proposal RAE.