DeCon: Path-based Conflict Detection Measurement for Knowledge Graph

Yihan Wang[1], Qi Song[1,2], Ling Zheng[1], Xiang-Yang Li[1,2]
[1] University of Science and Technology of China, Hefei, China
[2] Deqing Alpha Innovations Institute, Huzhou, China
yihanw@mail.ustc.edu.cn, qisong09@ustc.edu.cn, lingzheng@mail.ustc.edu.cn, xiangyangli@ustc.edu.cn

***Abstract*—Knowledge graphs (KGs), which utilize graph structures to represent real-world facts and associations, are pivotal in information retrieval and artificial intelligence. However, the necessity of constant updates often leads to errors or conflicts. Some conflicts, such as expression ambiguities, are challenging to detect at triple granularity and require semantic analysis together with other triples. In this paper, we define *path-based conflict* in KGs, extending error detection from triple granularity to path granularity. We demonstrate that path-based conflicts are confined to ring structures in KGs. We prove that indicating whether a triple brings such conflicts only needs to deal with anyone of its related ring structures, which significantly simplifies our problem. We propose RingE, a novel ring embedding method, to detect conflicts, and DeCon, a conflict detection framework suitable for both static and incremental scenes. This framework divides the path-granularity conflict detection task in KG into ring discovery and conflict detection. We also design a pruning strategy for ring discovery to improve the efficiency. Using real-world graphs, we experimentally verify that our algorithms are effective and feasible for large graphs. Our case study also verifies that the algorithm can detect real conflicts.**

***Index Terms*—knowledge graph, conflict, detection, ring**

I. Introduction

Knowledge graphs (KGs) are typically used to store and represent real-world information[1, 2]. They can significantly enhance the capability of data-driven applications across fields like data management[3, 4], information retrieval[5–7], and artificial intelligence. Notably, even today, when large language models (LLMs) are prevalent globally, KGs continue to demonstrate their vital role, particularly in addressing illusion issues and dealing with the problems for domain-specific knowledge[8–10]. Consequently, maintaining the integrity and quality of KGs is essential for executing downstream applications smoothly and correctly.

KGs use triples (h, r, t) to build networks that represent real-world knowledge. These KGs require continual updates and corrections to maintain accuracy and integrity[11]. Newly added knowledge may contain errors such as misunderstanding[12], which can adversely affect downstream applications if left uncorrected. For example, in malfunction detection for PCs, where the combinations of diverse software and hardware are complex, uncorrected errors or conflicts in newly added knowledge can hinder rather than help engineers[13, 14]. Due to the impracticality of manually verifying extensive triples, automated verification is essential to handling the large amount of data in KGs efficiently.

Error detection in KGs has been studied widely, which primarily uses learning-based methods and rule-based methods. Learning-based researches[12, 15, 16] mainly focus on discussing the correctness of a single triple and rely on the support of other triples for verification. In some scenarios, the semantics of each independent triple may not be directly considered to be right or wrong because it needs to rely on semantic connections with other triples in the specific KG. These researches mostly ignore errors caused by the semantic combination of complex relations that are associated with the relation path. However, while rules-based researches[17, 18] have begun to explore rules in paths, they mainly focus on the rules among the inherent attributes of the entities and relations, failing to pay enough attention to semantics. Such errors usually appear in the form of semantic conflicts and involve a series of interrelated triples, which are beyond the coverage of the rules for attributes. Consider the Exp.1.

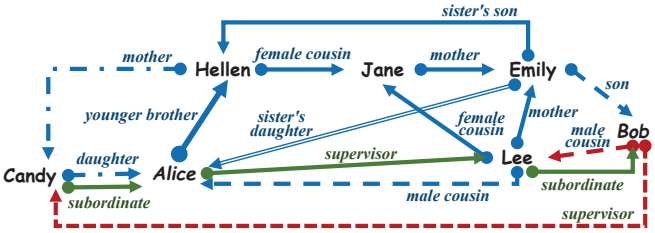


Fig. 1. Path-based error detection. We label the different types of relations with different colors: blue indicates kinship, green indicates working relations, and red indicates error. We also label the different rings for a unique triple with different lines: dotted lines, dashed lines, and double solid lines.

Example 1: Fig. 1 depicts relations and entities within a family where two potential conflicts exist. Our focus is on the first conflict involving Emily, Lee, and Bob. Individually, the relations $(Lee, mother, Emily)$, $(Emily, son, Bob)$, and $(Bob, male\ cousin, Lee)$ appear correct. However, there exists a conflict: both Lee and Bob have Emily as their mother, indicating that they are brothers, not cousins. In terms of semantic understanding, there are indeed differences between the expressions of cousin and brother, and this type of conflict is also a misunderstanding that often occurs in the real world. The same misunderstanding also appears in distinguishing cousins based on maternal or paternal lineage and considering age-based relations like older or younger siblings. □

As shown in Exp. 1, such conflicts may not be detected when just considered at triple granularity. However, if we take paths into consideration, these conflicts can be detected easily. In the scene of incremental KG construction, the newly added triples are likely to be correct individually, especially those for relations that have never appeared in the KG. But these triples may be semantically related to the triples in the existing KG. If these conflicts are not detected, they may lead to incorrect decisions in downstream applications. For example, in a KG of an enterprise's organizational structure, the same person cannot hold key roles in two unrelated departments at the same time. Detecting these conflicts is a critical step in ensuring the quality of the KGs. Consequently, there is an urgent need to design an error detection method for relations in the paths. The paths between two entities are usually complex, and learning the fixed rules from them will be challenging. Thus, we consider the learning-based method to detect conflict. However, some challenges still need to be resolved.

C1: The number of all paths in a KG is huge. The KG, seen as a directed graph, allows for varying numbers of paths between node pairs. The range of the number of possible paths extends from 0, indicating no connection, to an infinite count in the presence of cycles where nodes can re-encounter themselves via directed edges. Given $|V|^2$ potential node pairs in a graph with $|V|$ entities, if the average path count of node pairs is n, the number of all paths may be huge, which is $n \cdot |V|^2$. So, when detecting conflict, it is unrealistic to traverse all paths. Therefore, we need to find suitable representatives, which can cover all conflicts among paths in the KG.

C2: The length of the paths may be long. As mentioned earlier, paths containing loops may lead to repetition. Therefore, the segmentations before the first loop take on greater significance. Even excluding loops, the number of paths remains quite large, with a theoretical upper bound of $(|V| - 1)!$. Consequently, it is important for us to limit the maximum length of the paths so that they can cover as many conflicts as possible.

C3: The embedding and detection of path is different from triple. The process of embedding triples in KGs involves separately embedding the head entities, the tail entities, and the relations. Current KG error detection methods, which operate at triple granularity, require inputting each element of the triple and focus primarily on the semantics between the head and tail entities. In contrast, the path granularity methods demand a dynamic input due to variable path lengths. The semantic relation between a pair of nodes at path granularity remains constant, regardless of the path differences. Therefore, suitable embedding methods are required to accommodate varied paths between nodes, making conflict detection at path granularity significantly different from error detection at triple granularity.

Our work focuses on a novel problem as *path-based conflict* in KGs, extending the error detection problem from triple to path granularity. Our work is designed to detect more complex errors in KGs. In order to solve this problem, we propose a unique ring embedding method, RingE, for *detecting* conflicts within rings, along with a conflict detection framework DeCon.

Contributions. This paper studies path-based conflict detection algorithms in KGs.

- We define *path-based conflict* in KGs, an extension of error detection from the granularity of triples to that of paths. We theoretically prove that such conflicts are confined to ring structures within KGs.
- We have theoretically demonstrated that, in order to detect potential conflicts brought by a newly added triple, we only need to analyze one of its related ring structures. We propose a novel ring embedding method RingE, which uniquely combines head and tail entities with their relations and enables detection of conflicts within the KG.
- We propose DeCon, the first framework for complex conflict detection in KGs. This framework is divided into ring discovery and conflict detection, accommodating the requirements of both static and incremental scenes. For the static scene, DeCon implements a pruning strategy for ring discovery, significantly reducing the time complexity from $O(2^{|V|} \cdot (|V| + |E|))$ to $O(\frac{\zeta}{2\alpha} \cdot (|V| + |E|))$. This strategy improves the efficiency of detecting conflicts by up to 85% as verified in the experiment study.
- To evaluate our method, we design a conflict construction approach that injects conflicts into the rings in datasets. Then, We evaluate our work using three public datasets. The experimental results show that RingE effectively detects conflicts within ring structures with the accuracy of more than 90%, and DeCon has good performance where the time consumption is reduced by several magnitudes. We also show that DeCon can detect real conflicts in the case study.

These results yield effective methods toward conflicts detection in large graphs. All the proofs and complexity analysis can be found in [19].

Related Work. We categorize the related work into two parts: error detection in KGs and knowledge representation learning. We summarize these related works as follows.

Error Detection in Knowledge Graph. Error detection is an essential part of quality management in KGs. The main goal is to detect errors or outdated information[20], which can affect the reliability and usefulness of the KG[21].

Traditionally, detecting errors in KGs is done during their construction. But recently, more focus has been on detecting errors during the learning of knowledge representation. There are two main types of methods for this. First, there are embedding-based methods, like TransE[22] and its variations[23–25], ComplEx[26], and DistMult[27]. The second type is path-based methods [12, 15, 16, 28–30], which use the ideas of resource allocation[31] and confidence[32]. Most methods use TransE as the basic embedding method and add confidence framework in implementation. Several specific methods have been studied. For example, CKRL[16] uses the structure of the KG itself to find errors. SCEF[28] looks at both the KG structure and external text to fix errors. KGTtm[15] combines internal information with overall KG. PGE[12], for product KGs, integrates text and structure and adds triple

confidence to facilitate error detection. While, GAGED[30] creatively uses an unsupervised approach.

However, these methods mostly focus on simple errors at triple granularity in the KG and don't address complex relations formed by linking multiple triples. Therefore, errors and conflicts caused by relation derivations cannot be effectively detected.

In addition, some researches extend methods used in other types of entity resolution and conflict resolution methods to KGs[18, 33–40]. They use specific rules (like GFDs[39] and NGFDs[40]) and develop parallel algorithms[41, 42] to detect inconsistencies. These methods are more about mining and applying rules to detect errors on attributes of entities, while our problem focus on the semantics among them.

Knowledge Representation Learning. Representation learning involves the process of converting semantic information from images, texts, speech, and other mediums into low-dimensional, dense vectors, commonly referred to as embeddings. Knowledge representation learning (KRL) specifically deals with representing entities and relationships in KGs as such dense vectors. Traditional KRL methods, including the TransE series[22–25], ComplEx[26], DistMult[27], SimplE[43], and RotatE[44], vary in aspects such as representation space, scoring functions, coding models etc[45]. In recent years, the diverse range of downstream applications has necessitated knowledge representations that encapsulate entity semantics, thereby enhancing the effectiveness of KRL. This requirement has led to the development of various text-enhanced KRL methods[46–49]. For example, DKRL[48] employs entity description-enhanced representation learning, utilizing two distinct encoders to encode the semantics of entity descriptions.

However, most error detection methods primarily rely on traditional knowledge representation approaches. These approaches involve converting entities directly into vectors, learned exclusively from the triples in KGs. Thus, some researches have incorporated detailed descriptions of entities and relations in this process[12, 28]. However, it is essential to recognize that conflict detection involves semantic considerations. Therefore, in this study, we propose enhanced knowledge representation by integrating entity descriptions into the framework to improve its effectiveness.

II. Problem Formulation

We start with the notions of knowledge graphs.

Graphs. We consider a directed graph $G = (V, E, L)$, where V is a finite set of nodes and $E \subseteq V \times V$ is a set of edges. Each node $v \in V$ carries a label $L(v)$. In practice, the labels of the nodes and edges may represent types and relations (predicates), respectively.

Relation Path. When randomly selecting a head entity h and a tail entity t from the node set V, it is possible that there are various relation paths between them. Each path consists of an ordered sequence of several relations and entities that connect h to t, denoted as $p_{h\to t} = (r_1, e_1, r_2, e_2, \ldots, e_{m-1}, r_m)$. The two entities and the relations in the path can be represented as a path triple: $(h, p_{h\to t}, t) = (h, (r_1, e_1, r_2, e_2, \ldots, e_{m-1}, r_m), t)$. It is important to note that we do not allow loop in the path as Exp. 2. For each pair of head and tail entities, we can get a set of such paths, defined as:

$$P_{h\to t} = \{p^1_{h\to t}, p^2_{h\to t}, \ldots, p^l_{h\to t}\}. \tag{1}$$

Example 2: In Fig. 1, the path set from Alice to Bob, denoted as $P_{Alice\to Bob}$, includes the following two paths: $(younger\,brother, Hellen, female\,cousin, Jane, mother, Emily, son)$ and $(supervisor, Lee, subordinate)$. It is important to note that both paths contain a sequence of relations and entities that connect Alice to Bob without any entity repetition. However, while the path $(supervisor, Lee, male\,cousin, Alice, supervisor, Lee, subordinate)$ could also establish a connection from Alice to Bob, it involves a repetition of the entity "Lee", thus is not in $P_{Alice\to Bob}$. □

Conflict. There exist two path triples pertaining to entities h and t, specifically, $(h, p^u_{h\to t}, t)$ and $(t, p^v_{t\to h}, h)$. The two path triples are constructed using the same type of relations. Conflict means that the two path triples exhibit a significant semantic inconsistency. That is, among the sets $P_{h\to t}$ and $P_{t\to h}$, we can get a pair of paths denoted as $(p^u_{h\to t}, p^v_{t\to h})$ with manifest inconsistencies in their relations and the semantics conveyed by the involved entities.

Semantic Consistency. Following the ideas of TransE [50], for paths that are constructed using the same type of relations, semantic consistency is achieved when the sum of the relation vectors within path $p^u_{h\to t}$ is approximately equal to the negation of the sum of relation vectors within path $p^v_{t\to h}$.

$$\vec{r^u_1} + \vec{r^u_2} + \cdots + \vec{r^u_m} \simeq -(\vec{r^v_1} + \vec{r^v_2} + \cdots + \vec{r^v_n}) \tag{2}$$

Where m is the number of relations in $p^u_{h\to t}$, and n is the number of relations in $p^v_{t\to h}$.

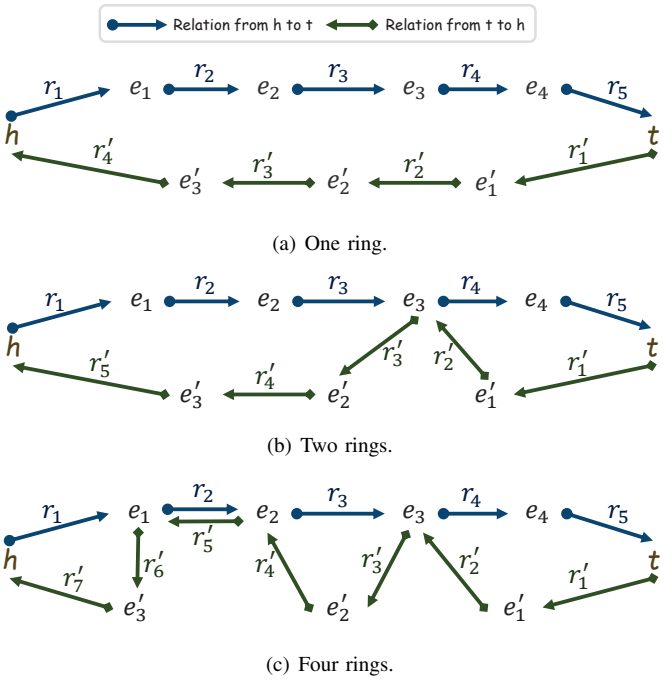
Semantic Inconsistency. Building upon the concept of semantic consistency defined earlier, semantic inconsistency is characterized by a scenario where the pair of paths $(p^u_{h\to t}, p^v_{t\to h})$ fails to meet the criterion of approximate equality outlined above.

$$\vec{r^u_1} + \vec{r^u_2} + \cdots + \vec{r^u_m} \neq -(\vec{r^v_1} + \vec{r^v_2} + \cdots + \vec{r^v_n}) \tag{3}$$

Example 3: Consider the path pair $\{(female\,cousin, Jane, mother), (sister's\,son)\}$ between Hellen and Emily in Fig. 1. From the first path, it becomes apparent that Hellen and Jane are cousins, implying that their respective mothers are different persons. Given that Jane's mother is Emily, it is suggested that Hellen may be the son of Emily's sister, not Emily. This pair of paths meets the criteria for semantic consistency, as the relations and semantics align logically. However, the path pair between Lee and Bob, $\{(mother, Emily, son), (male\,cousin)\}$, is semantically inconsistent. If we assume that the first path is

accurate, implying that both Bob and Lee are Emily's children, then the relation between Bob and Lee cannot be cousins. In fact, they must be brothers, making this path pair satisfying semantic inconsistency. Similarly, the path pair between Candy and Bob, $\{(subordinate, Alice, supervisor, Lee, subordinate), (supervisor)\}$, is also semantically inconsistent. □

Based on the previously defined concepts of relation path and conflict, we can derive the following *Insight* and *Theorem*.



(a) One ring.

(b) Two rings.

(c) Four rings.

Fig. 2. Examples of ring structures. Blue arrows indicate the path from head to tail, while green arrows indicate the path from tail to head.

Insight 1: *For any pair of relation paths, denoted as* $(p^u_{h\to t}$, *and* $p^v_{t\to h})$, *which connect the head entity* h *and the tail entity* t, *there exists at least a ring structurally.* □

Proof sketch: A pair of paths $(p^u_{h\to t}, p^v_{t\to h})$, connecting head entity h and tail entity t can be seen as a directed graph. Therefore, the path from h to t can be represented as $E = (e_1, e_2, \ldots, e_g)$, while the path from t to h is $E' = (e'_1, e'_2, \ldots, e'_q)$.

- If $\forall f \in (e_1, e_2, \ldots, e_g), f \notin (e'_1, e'_2, \ldots, e'_q)$, this situation can be abstracted as illustrated in Fig. 2(a). Consider any node k in the set of nodes N, there exists paths respectively from k to t in E, from t to h in E', and from h to k in E. The presence of a ring structure is confirmed.
- If $\exists f \in (e_1, e_2, \ldots, e_g), f \in (e'_1, e'_2, \ldots, e'_q)$, this can be abstracted as depicted in Fig. 2(b) and 2(c). In this case, let the set of nodes satisfying these conditions be represented as $F = (f_1, f_2, \ldots, f_j)$, maintaining the order in the path.
  - For the segment containing h, there exists paths from h to f_1 in E, and from f_1 to h in E'. As a result, a ring structure is established.
  - For the segment involving t, there exists paths from f_j to t in E, and a path from t to f_j in path E'. Thus, a ring structure is established.
  - For the remaining segments, arbitrarily chosen $f_s \in (f_2, \ldots, f_{j-1})$. There are paths connecting f_s to f_{s+1} in E, and f_{s+1} to f_s in E'. Consequently, ring structures are established.

To sum up, there must be at least one ring to form a pair of paths $(p^u_{h\to t}, p^v_{t\to h})$ between h and t. (detailed proofs could be found in [19].) □

Example 4: The path pair $\{(younger\,brother, Hellen, female\,cousin, Jane, mother, Emily, son), (male\,cousin, Lee, male\,cousin)\}$ in Fig. 1 connecting Alice and Bob conforms to the first situation outlined in the previous proof, which has only one ring. And for another path pair between Alice and Bob in Fig. 1, $\{(younger\ \ brother, Hellen, female\,cousin, Jane, mother, Emily, son), (male\,cousin, Lee, mother, Emily, sister's\ daughter)\}$, the same as Fig. 2(b), there exists structure with two rings within the path pair. □

Theorem 1: *Conflicts between the head entity* h *and the tail entity* t *must indeed exist within the rings of the path pair* $(p^u_{h\to t}, p^v_{t\to h})$. □

Proof: According to the above proof for Insight 1, a pair of paths $(p^u_{h\to t}, p^v_{t\to h})$ between h and t must be able to be divided into $(j+1)$ segments. Consequently, this pair of paths can be deconstructed into sets of sub-paths, each of which is demarcated by a common intermediary entity. These sub-paths are represented as follows: $< h \to f_1, f_1 \to h >, < f_1 \to f_2, f_2 \to f_1 >, < f_2 \to f_3, f_3 \to f_2 >, \ldots, < f_{j-1} \to f_j, f_j \to f_{j-1} >, < f_j \to t, t \to f_j >$.

Hence, these sub-paths can be represented as

$$\begin{cases} < f_{s-1} \to f_s, f_s \to f_{s-1} > & s \in [2, j] \\ < h \to f_1, f_1 \to h > & s = 1 \\ < f_j \to t, t \to f_j > & s = j. \end{cases}$$

We define the conflict detection function $De()$, which takes a path pair as input, and outputs conflict detection results (0 for no conflict, 1 for conflict). If it is known that there are conflicts in $(p^u_{h\to t}, p^v_{t\to h})$, then

$$De(p^u_{h\to t}, p^v_{t\to h}) =$$

$$Step[\textstyle\sum_{s=1}^{j-1} De(< f_{s-1} \to f_s, f_s \to f_{s-1} >) +$$

$$De(h \to f_1, f_1 \to h) + De(f_j \to t, t \to f_j)],$$

Where $Step[]$ is the step function in the activation function. In this definition, if it is known that there are conflicts in the given pair of paths $(p^u_{h\to t}, p^v_{t\to h})$, $De(p^u_{h\to t}, p^v_{t\to h})$ returns 1,

indicating the presence of conflicts. Otherwise, it returns 0 to signify the absence of conflicts.

To sum up, if it is known that there is a conflict in the pair of paths $(p^u_{h\to t}, p^v_{t\to h})$, there must be conflicts present in one or more of the sub-paths mentioned above. Conversely, if there is no conflict detected within any sub-paths, then the pair of paths $(p^u_{h\to t}, p^v_{t\to h})$ is considered to be conflict-free.

Therefore, conflicts are confined within the ring structure. □

Based on the preceding proof, it is established that all conflicts within a KG must inherently exist within the ring structures in the KG. Therefore, conflict detection can be performed by discovering and analyzing ring structures. In light of this, we propose the following definition for conflict detection within a KG.

Problem statement. Given a graph KG, the task is to discover all the ring structures within the KG and to analyze whether each of them meets the criteria for semantic consistency between the entities and relations forming the ring. Assuming that there exists a vector space for the representation of entities and relations, where entities are represented as $\{\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_w}\}$ and relations are represented as $\{\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_w}\}$, we can describe the ring structure as follows:

$$\emptyset verrightarrow\mathbb{R} \simeq \mathbf{e_1} + \mathbf{r_1} + \mathbf{e_2} + \mathbf{r_2} + \cdots + \mathbf{e_w} + \mathbf{r_w} \quad (4)$$

The problem of conflict detection can be represented as:

$$\overrightarrow{\mathbb{R}} \simeq \mathbf{e_1} + \mathbf{r_1} + \mathbf{e_2} + \mathbf{r_2} + \cdots + \mathbf{e_w} + \mathbf{r_w} \overset{?}{\simeq} \mathbf{0} \quad (5)$$

This problem is, nevertheless, nontrivial.

III. Detecting conflicts.

To detect conflicts within KGs effectively, we need to find a vector space and design an operator that satisfies the semantic relation. The comprehensive detection of conflicts within a KG necessitates an initial discovery of all rings within the graph. Thus, we need to address three hierarchical questions. 1) Is there a feasible approach to reduce the complexity of conflict detection within KGs? 2) What is the effective embedding method for the ordered relation paths to detect conflicts accurately? 3) How to use the conflict detection methods in real systems? We next answer these questions in this section.

Before proposing our methods, it is important to evaluate the feasibility and solvability of the conflict detection problem at first. If the complexity of conflict detection for a single triple is high, usability issues may arise, thereby potentially restricting the applicability to the entirety of the KG. To address this, we first explore strategies to effectively reduce complexity. This reduction is crucial for ensuring the practicality and scalability of our methods. Following this, we study the ring embedding method, designed specifically for conflict detection within KGs. Finally, we introduce a ring discovery algorithm and a real system framework for conflict detection. To facilitate real-world applications, we proposed algorithms for both static and incremental scene. As detecting rings on a large graph is a complex problem, we develop a pruning strategy to further reduce the complexity, thereby facilitating the efficient and effective application of our methods over large graphs.

A. Is there a feasible approach to reducing the complexity in conflict detection?

When considering whether a particular edge is associated with a conflict, it is necessary to analyze all rings that contain this specific edge. However, in cases where the length of the ring is not predefined, there is a possibility that the number of rings containing this specific edge is infinite. Even when the number of rings is finite, we need to do conflict detection for all the rings containing this specific edge. It is important to note that nodes and edges carry specific semantics, distinguishing them from corresponding nodes and edges in traditional graph structures. Notably, *relations of the same type within a KG share semantic relevance and can be logically derived, while relations of different types lack significant connections for meaningful discussion*. Based on the observation, we only need to focus on rings composed of relations of the same type. Nonetheless, we still have to traverse every ring.

Based on this understanding, we explore strategies aimed at mitigating the time complexity of the conflict detection problem. To achieve this goal, we have the following critical property and lemma, which may hold the key to achieving this reduction in time complexity.

Property. Since the semantic associations between two entities are fixed, and the semantic association of each type is unique, the paths between two entities formed by relations of the same type need to convey the same semantics. In other words, different paths with the same type of relations which connect the same pair of entities, should ideally express consistent semantics. This can also extend to the embeddings of these paths, where the results of operations performed on these embeddings should also be consistent.

Example 5: Fig. 1 presents two distinct types of relations: kinship, represented in blue, and working relations, depicted in green. We can easily find a path pair including both of them, like $\{(supervisor, Lee, mother, Emily, son), (supervisor, Candy, daughter)\}$ between Alice and Bob. In this example, it becomes evident that we cannot discuss the correctness of the relation between Bob and Candy. This is because kinship and working relations are fundamentally unrelated categories within this KG. Consequently, it is not logically sound to use information from the relations about kinship to infer or deduce the relations about working. □

Lemma 2: *To determine whether a specific edge in a KG is implicated in conflicts, assessing only one of all rings that include this edge is enough.* □

Proof: Given an edge $R_{A\to B}$ in a KG, we can find any two distinct paths from B to A in the KG, represented as

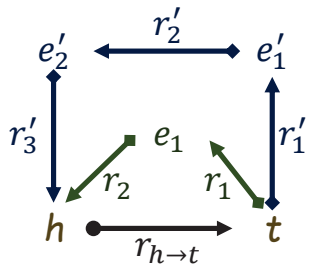


Fig. 3. An example of different rings with the same head and tail entity. For the relation from h to t, adding e_1 can get one ring (represented as green arrows with black one), while adding e_1' and e_2' can get the other ring (represented as red arrows with black one).

$p^1_{B\to A}, p^2_{B\to A}$. The proof for the Lemma from the perspective of triple is provided as follows, where $R_{A\to B} = (r_{AB})$,

$$p^1_{B\to A} = (r^1_1, r^1_2, \ldots, r^1_{i+1}), p^2_{B\to A} = (r^2_1, r^2_2, \ldots, r^2_{j+1})$$

Perform vector operations under ideal conditions, if there is a conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} = r^1_1 + r^1_2 + \cdots + r^1_{i+1} + r_{AB} \neq 0,$$

according to the above nature, there is $r^1_1 + r^1_2 + \cdots + r^1_{i+1} = r^2_1 + r^2_2 + \cdots + r^2_{j+1}$, and then

$$r^2_1 + r^2_2 + \cdots + r^2_{j+1} + r_{AB} = R_{A\to B} + p^2_{B\to A} \neq 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has conflict.

In the same way, if there is no conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} = r^1_1 + r^1_2 + \cdots + r^1_{i+1} + r_{AB} = 0,$$

according to the above nature, there is $r^1_1 + r^1_2 + \cdots + r^1_{i+1} = r^2_1 + r^2_2 + \cdots + r^2_{j+1}$, and then

$$r^2_1 + r^2_2 + \cdots + r^2_{j+1} + r_{AB} = R_{A\to B} + p^2_{B\to A} = 0.$$

That's to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has no conflict.

For extension, we also prove from the perspective of relation and using the idea of TransE, which means $h + r \simeq t$. The conclusion is also consistent with the above. (see [19] for detailed proof.) □

Building on the insights provided by the above Lemma, we only need to do conflict detection on any one of the related rings. This can significantly reduce the time cost of checking all related rings.

Example 6: In Fig. 1, for the triple $(Alice, younger\,brother, Hellen)$. This triple is part of three distinct paths that contribute to forming ring structures. They are delineated by different line shapes in the figure. These paths are $(mother, Candy, daughter)$, $(female\,cousin, Jane, mother, Emily, sister's daughter)$, $(female\,cousin, Jane, mother, Emily, son, Bob, male\,cousin, Bee, male\,cousin)$. For the purpose of conflict detection, it is not necessary to analyze all these paths. Instead, we can select any path, preferably the shortest. □

B. What is the effective embedding method for the ordered relation paths?

As outlined in Sec. I, traditional approaches in KRL begin by assigning a unique ID to each entity in the KG. The triples in the KG are then transformed into a list of these IDs, which are the input of the learnable embedding model. While such methods have effectively detected errors at *triple* granularity, they are unable to solve the problem of conflict detection at *path* granularity in the KG. For instance, the path embedding from head entity to tail entity and from tail entity to head entity should be complementary because they are semantically opposite. A critical limitation arises when considering the embeddings of paths. In an ideal scenario, embedding a path from a head entity to a tail entity and the reverse path from the tail entity to the head entity should be complementary, reflecting their semantically opposite nature. However, traditional KRL methods often fail to capture this nuance, leading to inadequate representation for the purpose of conflict detection at path granularity.

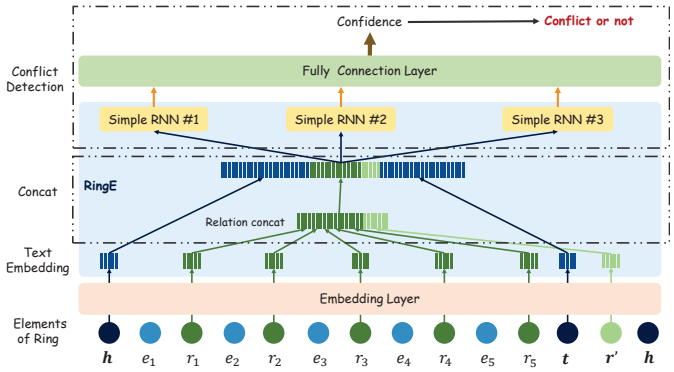


Fig. 4. RingE and Conflict Detection Models. The blue embedding vectors are for entities, and the green embedding vectors for relations. The three orange arrows form the ring embedding. The different part of the framework is gathering together, and RingE is gathering together with a blue background.

So, in our design, we propose a ring embedding method RingE, which employs the TransE model for initializing the embeddings of entities and relations within a ring structure. These initialized embeddings serve as the input for generating the ring embedding. Addressing the complexity of both individual entity and sequences of relations, we adopt the Simple RNN architecture as our baseline method. As depicted in Fig. 4, our method processes the embeddings of entities and their relation sequences, producing an output that represents the embedding of the ring. Following the proof in Sec. II, the relations in a path are more valuable that we can ignore the entities in the path and emphasize the significance of the relations in the path. Consequently, our method focuses primarily on the relations sequence, excluding the entities from the input sequence, which helps simplify the representation.

Then, we employ three distinct Simple RNNs, each with its unique number of unit, to generate three different outputs of the input sequence. These three outputs collectively constitute the embedding of the ring, which satisfies different concerns within the ring. The ring embedding, which consists of the outputs of the three RNNs, is then integrated using a full connection layer. The full connection layer synthesizes the three varied outputs and computes a confidence score. This score indicates potential conflicts within the ring, with lower scores suggesting a greater likelihood of conflict.

To further augment the conflict detection capabilities of our ring embedding method, we propose to replace the Simple RNN with more advanced sequence models, i.e., GRU and LSTM. These models are known for their enhanced capacity to process and interpret complex sequences. Furthermore, we also suggest replacing the TransE model with more advanced text embedding techniques, such as Bert and GPT-2. These models offer more robust mechanisms for capturing and representing the nuances of text. As illustrated in Fig. 4, our method, RingE, is designed to be flexible, allowing for integrating other advanced models. By replacing the text embedding and ring embedding modules with more powerful alternatives, the effectiveness of the method can be significantly enhanced. This adaptability ensures that RingE can progress with developments in model architectures.

Example 7: In Fig. 1, the embedding of the path $(Alice, younger\ brother, Hellen, female\ cousin, Jane)$ ideally exhibits a complementary relation to the embedding of the reverse path $(Jane, mother, Emily, sister's\ daughter, Alice)$. Similarly, the path embedding for $(Emily, son, Bob)$ should be complementary to that of $(Bob, male\ cousin, Lee, mother, Emily)$, so as the others. □

C. How to use the conflict detection methods in real systems?

The above method relies on the input of a known ring, thus the next question is how to find the ring. Given the potentially infinite number of rings of varying lengths within a KG, selecting appropriate rings as inputs for the algorithm becomes crucial. To address this challenge, we propose a comprehensive system framework specifically designed to adapt our methods for practical use in both *Static scenes* and *Incremental scenes*.

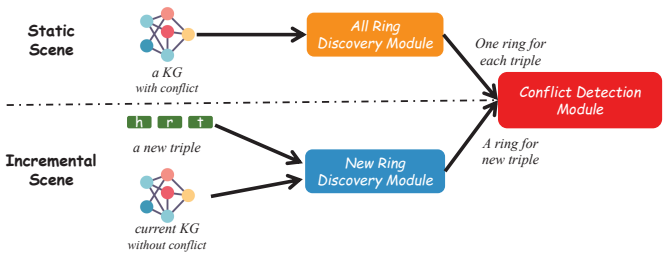


Fig. 5. Rind discovery under static and incremental scene.

As illustrated in Fig. 5, our framework comprises two principal modules: *Ring Discovery* and *Conflict Detection*. As detailed in Alg. 1 and Alg. 2, the ring discovery module identifies potential conflict-inducing rings within the KG. Due to different scene settings, this module is applied differently in static and incremental scenes. We implement a ***pruning strategy*** to reduce the complexity in static scenes. This strategy effectively reduces the time complexity from $O(2^{|V|} \cdot (|V|+|E|))$ to $O(\frac{\zeta}{2\alpha} \cdot (|V|+|E|))$. Here, ζ represents the upper bound of the number of triples appearing in the rings. The conflict detection module, as described in Alg. 3, employs the DeCon method to analyze whether a conflict exists in the discovered ring. These modules form a system that detects conflicts within KGs.

Algorithm 1: Ring Discovery(Incremental)

Input: T_{KG} :a set of triples from one KG without conflict, $tri_{new} =< h, t >$:a triple to be added in the above KG, τ: lower bound of the number of relations in a ring, ζ: upper bound of the number of relations in a ring.

Output: γ: a ring contain t_{new}.

```
Initialize a stack S with h and t.
while len(stack) ≤ ζ do
    if len(stack) ≥ τ − 1 then
        Find the neighbor entities NE for t.
        if h in NE then
            if len(stack) ≥ τ then
                Break.
        else if h not in NE then
            Do one step DFS and find a new node n'.
    if n' not in S then
        Add a new node to S.
    else if n' is in S then
        if n' is h && len(stack) ≥ τ then
            Find the corresponding triples γ for all the
             nodes in S.
            Break.
return γ.
```

*1) **Ring Discovery**:* When executing the ring discovery algorithm in a directed graph, determining whether a ring exists carries a time complexity of $O(|V| + |E|)$, where $|V|$ is the number of nodes, and $|E|$ is the number of edges in the graph. However, when dealing with dense graphs where it is necessary to check for edges between each pair of vertices, the time complexity can approach $O(|V|^2)$. And then, finding all rings in this KG can lead to a near-exponential number of rings, potentially close to $2^{|V|}$ in worst case where every node in the KG is potentially connected to every other node. For many graphs, in the average cases, time complexity may be much lower than the worst case, but may still be exponential because the number of rings may grow rapidly. The task of discovering all rings in a KG thus have a time complexity of $O((|V| + |E|) \cdot 2^{|V|})$, and $O(|V|^2 \cdot 2^{|V|})$ in

dense graph. Therefore, if the number of relations in the ring is not specified, according to the Lemma 2 in Sec. III-A, even the seemingly simple task of discovering a corresponding ring for each edge can approach a time complexity of $O(|V|^2)$. Additionally, the significance of a path tends to diminish with increasing length, suggesting that excessively long paths may be meaningless. Hence, limiting the number of relations in a ring becomes imperative. To address this, we introduce two parameters: τ as the lower bound and ζ as the upper bound of the number of relations in a ring.

Algorithm 2: Ring Discovery(Static)

Input: KG :a knowledge graph with a set of triples $Tri = \{tri_1, tri_2, \dots\}$ with conflict, τ: lower bound of the number of relations in a ring, ζ: upper bound of the number of relations in a ring.

Output: $\Gamma = \{\gamma_1, \gamma_2, \dots\}$: a set of rings that may contain conflicts.

1 Initialize a boolean dict $visited$ (the keys are T, and the values are $False$ at initial), a stack S, and a result list Γ.
2 **for** *each* tri_i *in* T **do**
3 **if** $visited[tri_i]$ *is* $True$ **then**
4 Break.
5 Add the head entity h_i and tail entity t_i to S.
6 **while** $len(stack) \leq \zeta$ **do**
7 Do one step DFS and find a new node n'.
8 **if** n' *not in* S **then**
9 Add a new node to S.
10 **else if** n' *is in* S **then**
11 **if** n' *is* h *and* $len(stack) \geq \tau$ **then**
12 Find the corresponding triples γ for all the nodes in S.
13 Modify the values to $True$ in $visited$ for all the triples in γ.
14 Add γ to Γ.
15 Make S and γ empty.
16 Break.
17 **return** Γ.

Incremental scene. We will first show our design for incremental scenes, which is much simpler. In this scene, we consider a conflict-free KG with a new triple, which will be added. This new triple, not previously existing in the KG, is defined by its head entity h and tail entity t, as outlined in Alg. 1. The algorithm is initialized with h as the starting node and t as the secondary node in a deep first search (DFS) algorithm. The goal for the incremental scene is only to discover one ring containing the new triple, and we choose the shortest ring to minimize time consumption. Consequently, if the goal is only to discover the shortest ring, the time complexity is $O(\zeta n)$, in which n is the number of triples of the neighbors. In the worst-case scenario, especially in a dense graph where the neighbor count of t approaches $|V|$, the time complexity may rise up to $O(\zeta|V|)$. That is to say, by the Lemma 2 in Sec. III-A, we reduce the time complexity from $O(|V|^2)$ to $O(\zeta|V|)$ in the worst case. Furthermore, the time complexity in an ideal scene is $O(\zeta)$ with only one neighbor for each triple.

Static scene. As for the static scene, where the goal is to conduct ring discovery for each triple in the KG, the approach differs from the incremental scene. As the incremental scene demonstrates, the worst-case time complexity for ring discovery is $O(\zeta|V|)$. Extending this process to every triple in the KG, the time complexity could rise up to $O(\zeta(|V| + |E|))$, $O(\zeta|V|^2)$ in dense graph. We introduce a pruning strategy based on the Lemma 2 in Sec. III-A to optimize this process. This strategy requires only one ring for each triple to conduct conflict detection effectively. However, a single ring often encompasses multiple triples. Therefore, as Alg. 2 demonstrates, we label these triples (as per Line 13 of the algorithm) to avoid redundant ring discoveries (Lines 3 ˜4). With this pruning strategy, the time complexity can be reduced to $O(\frac{\zeta}{2\alpha} \cdot (|V| + |E|))$, where $\alpha \in [1, |V|]$ represents the average number of rings that a triple exists in. Consequently, when $\alpha = |V|$, the time complexity can be $O(\frac{\zeta}{2\alpha} \cdot (1 + \frac{|E|}{|V|}))$, but when $\alpha = 1$, the time complexity is $O(\frac{\zeta}{2} \cdot (|V| + |E|))$. In dense graph, the time complexity is $O(\frac{\zeta}{2}|V|^2)$, and when$\alpha = |V|$, it approximates $O(\frac{\zeta}{2}|V|)$.

*2) **Conflict Detection**:* The task for this module is conflict detection. We employ the embedding and detection model detailed in Sec. III-B to detect conflicts within a ring. As shown in Alg. 3, the input is a ring, and the output is whether the ring have conflicts. To analyze a set of rings, Alg. 3 can be repeated as many times as the number of rings.

Algorithm 3: Conflict Detection

Input: γ: a ring.

Output: χ: the result of conflict or not.

1 Put γ in the conflict detection model in Sec. III-B.
2 Get the result χ of γ.
3 **return** χ.

IV. Evaluation

Using three real-world graphs, we will next experimentally verify the effectiveness and efficiency of our methods for detecting conflicts.

Experiment Setting. We used the following setting.

Datasets. In this paper, we evaluate DeCon over three real-world datasets: WN18RR[51], NELL-995[52] and YAGO 3-10[53]. WN18RR is the subset of WordNet[54], and is widely used in the mistake detection in triple scenery. The details about the datasets are summarized in Tab. I.

Conflict Construction. However, there are no labels about conflict in any of the above datasets. Therefore, we propose a conflict construction method to inject conflict into these

TABLE I
DATASET STATISTICS

Dataset	#Entities	#Relations	#Relation types	#Rings (length from 3 to 5)			#Positive Rings	#Negative Rings
				#Rings (length 3)	#Rings (length 4)	#Rings (length 5)		
WN18RR	40943	11	1	8862 / 17724	28608 / 57216	28040 / 56080	65510	65510
NELL-995	75492	200	10	2362* / 4724*	12192* / 24384*	70328* / 140656*	84882*	84882*
YAGO 3-10	123182	37	3	40754* / 81508*	33909* / 67818*	36197* / 72394*	110860*	110860*

* The number of rings for YAGO 3-10 is only the number we used in our experiments, in fact, the number is very large.
/ The number before slash indicates the rings without conflict, which comes from the ring discovery. The number after slash indicates the rings with conflict, which is the output of our conflict construction method.

datasets to simulate conflicts in KG. Inspired by the negative triples construction method in CKRL[16], we construct datasets with labels about conflict. Specifically, given a ring $(e_1, r_1, e_2, r_2, \ldots, e_w, r_w)$ with the same type of relations, in which r_w is the relation from e_w to e_1, we randomly replace one relation or more in the ring with the same type of relation. Thus we can get a ring with conflict $(e_1, r_1, \ldots, r'_{false}, \ldots, e_w, r_w)$. As Sec. III-A said, the rings with the same type of relations are useful, so we need to classify the relations for these datasets.

Unfortunately, the relations in these three datasets do not have any labels, so we need to classify the relations in another way. We do use unsupervised clustering methods to classify relations and use the clustering results as classification criteria. However, we have observed that the classification results are not satisfactory. So, we manually classify the relations and divide the relations in NELL-995 and YAGO 3-10 into several types according to the type of the tail entities which may be connected to the relation. The sub-graphs of some relation types contains too many rings (we did not finish ring discovering for 48 hours), so we only used some of the relation types in our experiments. After classifying and doing ring discovery on these three datasets, we can summarize the datasets in Tab. I.

Algorithms. We implemented DeCon-simple, DeCon-GRU and DeCon-LSTM in Python using Keras[55] and compare it with the following baseline and state-of-the-art (SOTA) algorithms.

1) TransE [22]: The most classic algorithm in structure-based KG embedding, which uses the idea of $h + r \simeq t$.
2) DistMult[27]: Another classic algorithm in structure-based KG embedding, which concentrates on mining logical rules and capturing relational semantics.
3) KGTtm[15]: The classic error detection algorithm at triple granularity, which uses a crisscrossing neural network structure to integrate internal semantic information and KG global reasoning information to quantify the semantic correctness of triples and the authenticity of the facts expressed.
4) CAGED[56]: The SOTA error detection algorithm at triple granularity, which is an unsupervised learning mechanism integrating contrastive learning and KG error detection. CAGED augments a KG into different hyper-views by using each triple as a node to detect errors.

Implementation. Our goal is to detect the conflict in a ring within the KG. According to the above, only rings with limited length may make sense. In practice, we choose 3 as the lower bound and 5 as the upper bound. That is, the length of our rings in the experiment is $[3, 5]$. And the length is the number of triples in the ring, which equals the number of relations in the ring. But is not the length of the path list, which may contain the entities in the path. The length of the path list may vary from 6 to 10 when it contains both entities and relations.

We implement our embedding model using Keras, which has already implemented the RNN methods and full connection layer for users. The dimension of the entity and relation embeddings for the input of the model is 100, and we use TransE as the initialization embedding method. Among the models, the batch size is fixed to 50, and we use three RNN modules with 50, 80 and 100 units. The stop conditions for model training are not only determined by the max number of epochs, which is 200, but also is affected by the early stopping method[57] based on the performance on the valid datasets. We use squared hinge loss[58], which performs better in our two-category task. Adam optimizer is used as the parameter optimization, which is commonly used. To reduce overfitting, we use the dropout method. Specifically, our inputs of the embedding model contain a relation list, which indicates the ring, so we use the mask and global average pooling to deal with the dynamic length of the relation list.

We apply the same settings to all baseline methods to make the comparison fair. We ran all our experiments on a Linux machine powered by an NVIDIA GeForce RTX 4090 GPU, a 32-core Intel 3.0 GHz CPU with 128 GB of memory. We ran each experiment 10 times and report the averaged results.

Experimental results. We next present our findings. For more experiments, please refer to the full version[19].

Exp-1: Validity of Conflict Detection. We evaluate our conflict detection model with the above baseline methods in terms of accuracy, which is the core of our work. In order to comprehensively evaluate our proposed model framework, we compare DeCon with simple RNN, GRU and LSTM, which are all implemented based on Keras. Worth mentioning that the baseline methods are all designed for error detection at triple granularity, we modified r in the input to a relation list R, and using "GlobalAveragePooling1D" in Keras to compress the shape of R to r.

The evaluation results are presented in Tab. II. Our model framework can effectively detect conflicts. The result shows that DistMult cannot fit the demand of error detection at path granularity, tending to predicting all test data to be negative. In order to follow the conflict construction strategy of error detection at triple granularity, which set the ratio of positive and negative samples to 1 : 1, we also let the ratio of positive and negative samples in test data be 1 : 1. Thus, the result is 0.5, for all the positive samples also predicted to be negative. We notice that the most classic method, TransE, shows better performance. That is because our theoretical basis and text embedding method are both based on TransE. The performance of TransE is close to our detection model when following our method, because we replace the simple RNN module in the framework with TransE directly in the experiment. TransE performs best among all the baselines. Unfortunately, the error detection methods at triple granularity, which are designed for triples, show low accuracy close to 0.5. Our methods performs better than all baselines including TransE. All three methods achieve best accuracy higher than 90%, representing a significant improvement compared to other methods.

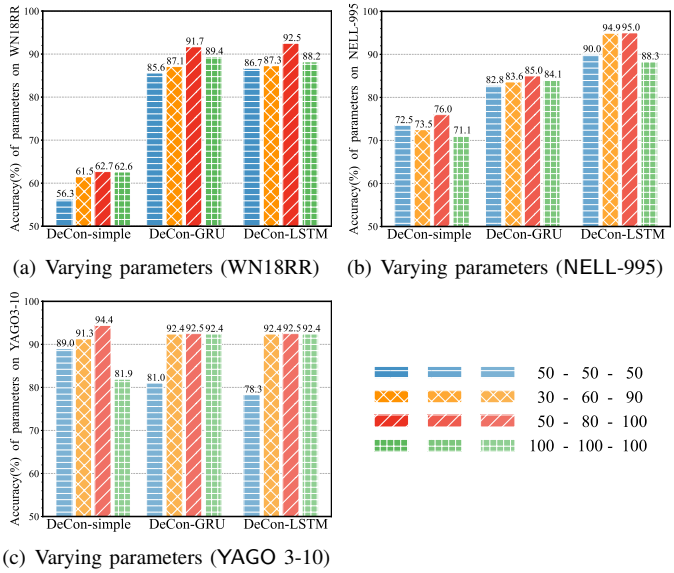
TABLE II
COMPARISON OF AVERAGE ACCURACY (ACC) PERFORMANCE AMONG DIFFERENT METHODS.

Types	Method	WN18RR	NELL-995	YAGO 3-10
Traditional Methods	TransE [22]	0.883	0.900	0.881
	DistMult[27]	0.500	0.500	0.500
Triple Error Detection	KGTtm[15]	0.500	0.500	0.500
	CAGED[56]	0.503	0.499	0.499
Our Methods	DeCon-simple	0.627	0.760	**0.944**
	DeCon-GRU	0.917	0.850	0.925
	DeCon-LSTM	**0.925**	**0.950**	0.925

The bold numbers represent the best performances among all.
The underlined numbers represent the best performances among all baselines.

Among the three network structures, we find out that LSTM shows better results than GRU, especially over WN18RR and NELL-995. LSTM shows a greater performance improvement compared to GRU and simple RNN, which can be improved by more than 30% for WN18RR and close to 20% for NELL-995. Compared with the previous error detection method, DeCon shows better detection capabilities for complex conflicts, and the accuracy exceeds 90% in most cases. We also notice that for the cases where all the relations in the rings have the same type, our methods can achieve close to 100% accuracy .

Varying model parameters. We also explored different model parameter selections to study the effects of the parameter selection on the performance of our model framework. We selected 3 additional sets of parameters to compare with our parameters, respectively set as $50-50-50$, $30-60-90$ and $100-100-100$, which has the same order of magnitude as the embedding dimension of TransE. As shown in Fig. 6, our selections of parameters can meet the best among them. That's



(a) Varying parameters (WN18RR) (b) Varying parameters (NELL-995)

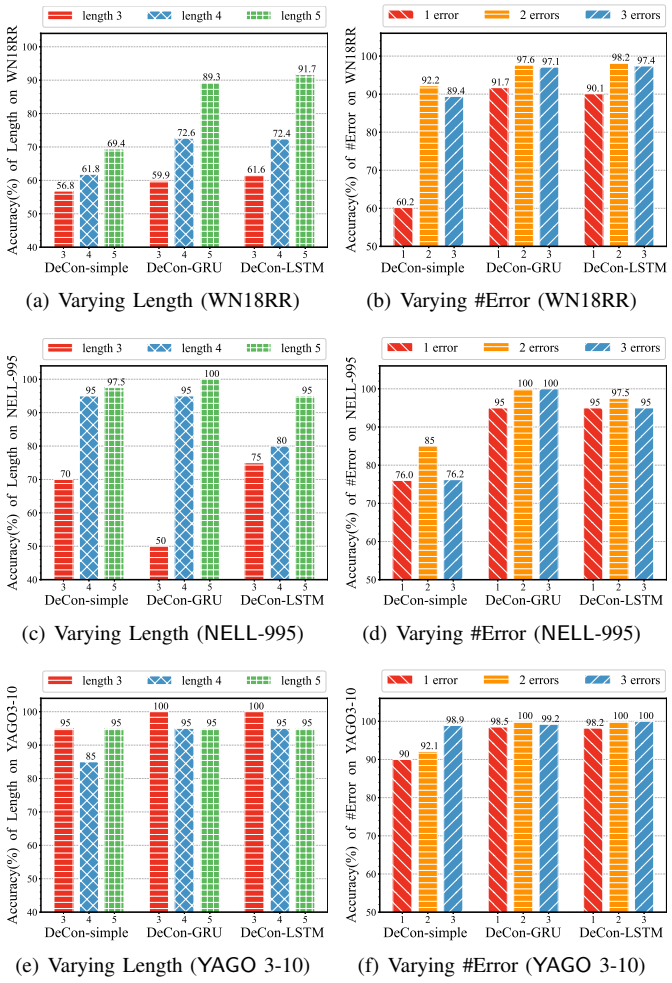
(c) Varying parameters (YAGO 3-10)

Fig. 6. Results of conflict detection with different number of units.

because different number of units among different models can make the output different, and the irregular number of units selected can make the difference more obvious.

Varying path length. We evaluate effect of the length of the path based on accuracy, as shown in Fig. 7(a), 7(c) and 7(e). The length of the path indicates the number of relations in the path. We conduct experiments on length 3, 4 and 5 respectively, and set the same number of conflict triples for each path, which is 1 in experiment. The result shows that, within a reasonable range, the longer the path is, the more evidence there is and the higher the accuracy of the detection is. For length 3, we notice that the accuracy is much smaller than length 4 and 5, that's because of the datasets. In order to minimize the impact of manually injecting conflicts to construct the datasets, the datasets we used to test different path lengths were extracted from the overall experiment according to the length of the path. As Tab. I shows, the number of paths of length 3 is much smaller compared with those of length 4 and 5, so the model for length 3 may not fit well enough. But this does not prevent us from drawing the above conclusions.

Varying error rate. We next evaluate effect of the number of errors in the path and the results are shown in Fig. 7(b), 7(d) and 7(f). We still use all paths with length 3 to 5, but we increase the number of errors for them. Based on the original setting which includes only single error, we modified 2 triples and 3 triples to construct the triple conflict and conducted the experiment. It is worth noting that, since the length of the path is 3, the number of error triples is at most 3 under our experimental setting. The results show that, the more error triples exist in the path, the more obvious the conflict will be and the easier it will be detected. However, since the length of the path we used is 3 to 5, when the conflict is 3, a *pseudo-replacement* is performed for the path with length 3. After replacement, there is a probability that the path is conflict-



(a) Varying Length (WN18RR) (b) Varying #Error (WN18RR)

(c) Varying Length (NELL-995) (d) Varying #Error (NELL-995)

(e) Varying Length (YAGO 3-10) (f) Varying #Error (YAGO 3-10)

Fig. 7. Results of Conflict Detection with different path lengths and error numbers. (a), (c) and (e) show the effect of the length of the path while the number of the error triples are fixed. (c), (d) and (f) show the effect of the number of the error triples in the path, while the length is from 3 to 5, which is the same as the experiments in Tab. II. (a) and (b) are experiments on WN18RR. (c) and (d) are experiments on YAGO 3-10. (e) and (f) are experiments on NELL-995.

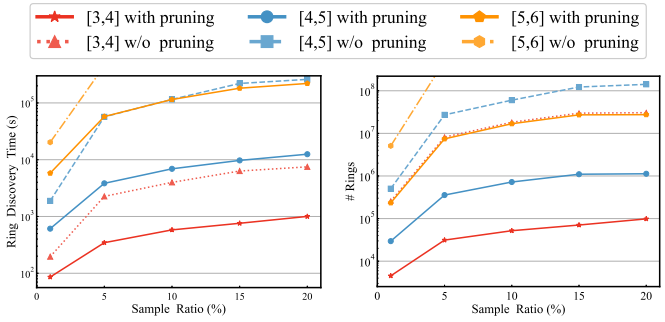
free, so it will mislead the model. Thus, when the number of conflict injection is 3, the accuracy decreases slightly.

Exp-2: System Efficiency. We then evaluated the efficiency of the system in static scene. As we mentioned above, YAGO 3-10 has a large amount of rings that cannot be collected within 2 days. So YAGO 3-10 has enough rings for us to evaluate in this part, and we use the sub-graphs of YAGO 3-10 to evaluate the efficiency under different graph sizes. We use a sample strategy to get the sub-graphs, in which we randomly choose a set of entities in YAGO 3-10 following the sample ratios and then we do breath first search (BFS) for all of these entities to get all the 3 hops entities and relations among them. Through pre-experiments, we find that when the sampling rate is 20% with $50k$ entities and $220k$ triples, and mining rings with a length of 3 to 5, the pruning strategy consumes 3.5 hours discovering $1640k$ rings, while the program doesn't end within 48 hours without pruning. So in our experiments we set the sample ratio as 1%, 5%, 10%, 15% and 20%, the number of entities and triples in these sub-graphs are summarized in Tab. IV. And we evaluate the efficiency with the lengths of $[3, 4]$, $[4, 5]$ and $[5, 6]$. It's worth state that for lengths $[5, 6]$, only 1% without pruning can get the result within 48 hours, so we add 1% to our results. We compare DeCon with pruning and without pruning to demonstrate the effectiveness of our pruning strategy. The efficiency improvement on NELL-995 is similar to that of YAGO 3-10. Pre-experiments with the sample ratio as 1% and the length as $[3, 4]$ only output 3 rings on WN18RR. And the number of rings in WN18RR with length $[3, 5]$ is only $60k$, it doesn't has enough rings for this experiment. For result in NELL-995, please refer to [19].

TABLE III
THE STATISTICS OF SAMPLE RESULT OF YAGO 3-10.

Sample Ratio(%)	# Entities	# Triples
1	13999	32917
5	28899	89309
10	39383	139399
15	47174	187602
20	53872	229133

The results in Fig. 8(a) and 8(b) show our pruning strategy can greatly reduce the time consumption of ring discovery, and then, can greatly reduce the number of rings for subsequent conflict detection task. It is worth mentioning that since our pruning strategy for ring discovery is specific to our tasks and based on our derived Lemma, we only study ring discovery methods that find all rings.



(a) Time consumption. (b) The number of rings.

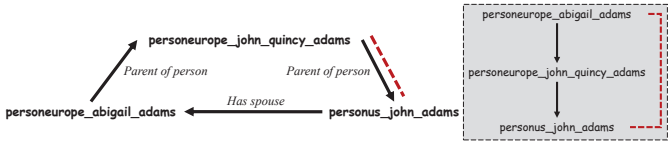
Fig. 8. Results of Ring Discovery with and without pruning. (a) shows the time consumption with and without pruning. (b) shows the number of rings with and without pruning. The results out of the figure are those who cannot get the result within 48 hours. The same color with different shades is for the same path length, the dark color represents with pruning, and the light color represents without pruning. The all solid are the results with pruning.

In Fig. 8, we notice that there are some nodes out of figures, because we cannot complete the ring discovery program during 48 hours, which means the actual time consumption and the number of rings may be very large. The results of ring discovery without pruning are all above that of discovering with pruning. The time consumption can be improved by up to 85% for $[3, 4]$, up to 92% for $[4, 5]$, close even up to 100%

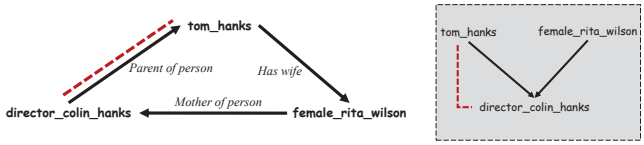
for $[5, 6]$ of sample ratio 5%, 10%, 15% and 20%. The higher the sample ratio, the more time consumption reduction. And the longer the length of the ring, the more time consumption reduction. The results can meet our analysis in Sec. III-C. The results show that the pruning strategy based on Lemma 2 can significantly reduce time consumption. Since fewer rings need to be detected, DeCon can certainly further reduce the conflict detection time for static scene.

Exp-3: Case Study.

The above experiments have shown that RingE performs well for conflict detection, and DeCon has a strong ability to detect conflicts at path granularity. We next conducted a case study to further demonstrate the ability of RingE and DeCon in detecting complex conflicts at path granularity in the real world. We gave examples of detecting errors on NELL-995 as Fig. 9 and 10 shown. Note that these conflicts can not be detected by any baselines including TransE.



(a) case 1

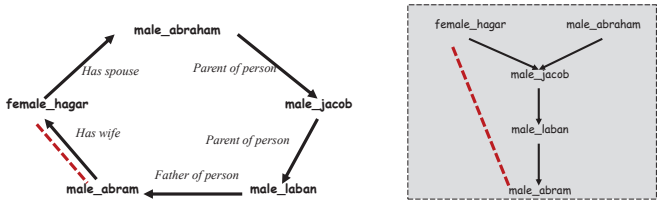


(b) case 2

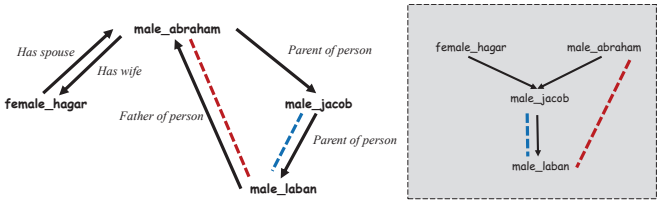
Fig. 9. Case 1 and 2. The left graph in each figure is the sub-graph in the NELL-995, and we rewrite it as a kinship tree on the right of each figure.

In order to make the kinship clearer, we rewrite it as a kinship tree on the right of the figures. We have observed semantic contradictions in the relations making up certain path, leading to the path been classified as conflict. In Fig. 9, the conflicts in the two cases share the same form and are both caused by the misdirection of the edge denoting the relation in a triple. Our method can catch the semantic conflict that a person's children cannot be his or her parents. Because the data in NELL-995 are all from the real world web, we can search the web to confirm the results. For case 1 in Fig. 9(a), we confirm that "personeurope_abigail_adams" and "personus_john_adams" are couples, and are parents of "personeurope_john_quincy_adams". Thus, the relation between "personeurope_john_quincy_adams" and "personus_john_adams" is incorrect, which causes the conflict, and the relation must be reversed. Similarly for case 2 in Fig. 9(b), we confirm that "tom_hanks" and "female_rita_wilson" are couples, and "director_colin_hanks" is their child. So, the relation between "director_colin_hanks" and "tom_hanks" must also be reversed.

In Fig. 10, the conflict is more complex, and we will explain it step by step. Before explaining, We need to make it clear that in NELL-995, "parent" refers to other elders besides mother and father. First, our method catches the semantic conflict in the ring, and we notice that "female_hagar" has two husbands: "male_abraham" and "male_abram". According to the kinship tree, there is a three-generation gap between the two people, so there may be a conflict. Then, we confirm that "male_abraham" and "male_abram" refers to the same person who changed his name. Furthermore, we can align these two entities as Fig. 10(b) shows. Afterward, our method still catches a conflict in the newly generated rings that the parent of the parent of the person cannot be the child of the person. Furthermore, we confirm that the relation between "male_abraham" and "male_laban" must also be reversed. Furthermore, we notice that "male_jacob" is the grandson of "male_abraham", and "male_laban" is the maternal uncle of "male_jacob". Thus, the relation between "male_jacob" and "male_laban" must also be reversed.



(a) step 1



(b) step 2 and 3

Fig. 10. Case 3. (b) red lines indicate step 2 and blue lines are step 3.

The above conflicts are all difficult to detect at triple granularity, but can be easily detected at path granularity.

V. Conclusion

In this paper, to eliminate the problem of semantic inconsistency in multi-triplet concatenation, we discuss path-granularity conflicts in KG. We theoretically prove that these conflicts are associated with ring structures. Furthermore, we simplify the complexity of conflict detection. We propose a ring embedding method RingE to vectorize ring structures and design DeCon, a conflict detection framework based on RingE. We designed a conflict detection framework on KG applied to static and incremental scenes respectively. Based on theoretical proof, we propose a pruning strategy to reduce the time consumption of the algorithm. Extensive evaluation with three real-world datasets shows that RingE can accomplish the conflict detection task well, and the pruning strategy in DeCon can exponentially reduce the time consumption and the number of rings used for detection. Future work includes model optimization for detection task and employing complex language models for ring representation.

REFERENCES

[1] Google, "Introducing the knowledge graph: things, not strings." https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012. Access: 2023-11.

[2] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, A. Wahler, D. Fensel, *et al.*, "Introduction: what is a knowledge graph?," *Knowledge graphs: Methodology, tools and selected use cases*, pp. 1–10, 2020.

[3] Y. Sun, J. Yu, and M. Sarwat, "Demonstrating spindra: A geographic knowledge graph management system," in *IEEE ICDE*, pp. 2044–2047, IEEE, 2019.

[4] S. Yu, Z. Yuan, J. Xia, S. Luo, H. Ying, S. Zeng, J. Ren, H. Yuan, Z. Zhao, Y. Lin, *et al.*, "Bios: An algorithmically generated biomedical knowledge graph," *arXiv*, 2022.

[5] A. Asai, S. Min, Z. Zhong, and D. Chen, "Retrieval-based language models and applications," in *ACL*, pp. 41–46, 2023.

[6] Y. He, Q. Jia, L. Yuan, R. Li, Y. Ou, and N. Zhang, "A concept knowledge graph for user next intent prediction at alipay," *WWW*, 2023.

[7] D. Zheng, X. Song, C. Ma, Z. Tan, Z. Ye, J. Dong, H. Xiong, Z. Zhang, and G. Karypis, "Dgl-ke: Training knowledge graph embeddings at scale," in *ACM SIGIR*, pp. 739–748, 2020.

[8] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *arXiv*, 2023.

[9] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, and M. Martin, "Llm-assisted knowledge graph engineering: Experiments with chatgpt," *arXiv*, 2023.

[10] Z. Zhang, Z. Zeng, Y. Lin, H. Wang, D. Ye, C. Xiao, X. Han, Z. Liu, P. Li, M. Sun, *et al.*, "Plug-and-play knowledge injection for pre-trained language models," *arXiv*, 2023.

[11] B. Xue and L. Zou, "Knowledge graph quality management: a comprehensive survey," *TKDE*, 2022.

[12] K. Cheng, X. Li, Y. E. Xu, X. L. Dong, and Y. Sun, "Pge: Robust product graph embedding learning for error detection," *VLDB*, vol. 15, no. 6, p. 1288–1296, 2022.

[13] H. Han, J. Wang, X. Wang, and S. Chen, "Construction and evolution of fault diagnosis knowledge graph in industrial process," *TIM*, vol. 71, pp. 1–12, 2022.

[14] H. Liu, R. Ma, D. Li, L. Yan, and Z. Ma, "Machinery fault diagnosis based on deep learning for time series analysis and knowledge graphs," *Journal of Signal Processing Systems*, vol. 93, pp. 1433–1455, 2021.

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

[16] R. Xie, Z. Liu, F. Lin, and L. Lin, "Does william shakespeare really write hamlet? knowledge representation learning with confidence," in *AAAI/IAAI/EAAI*, AAAI Press, 2018.

[17] W. Fan, W. Fu, R. Jin, M. Liu, P. Lu, and C. Tian, "Making it tractable to catch duplicates and conflicts in graphs," *PACMMOD*, vol. 1, no. 1, pp. 1–28, 2023.

[18] W. Fan, C. Tian, Y. Wang, and Q. Yin, "Parallel discrepancy detection and incremental detection," *VLDB*, vol. 14, no. 8, pp. 1351–1364, 2021.

[19] "Full version." https://yolandawang03.github.io/icde2024full.pdf. 2023.

[20] Y. Cheng, L. Chen, Y. Yuan, G. Wang, B. Li, and F. Jin, "Strict and flexible rule-based graph repairing," *TKDE*, vol. 34, no. 7, pp. 3521–3535, 2022.

[21] A. Zaveri, D. Kontokostas, S. Hellmann, J. Umbrich, M. Färber, F. Bartscherer, C. Menne, A. Rettinger, A. Zaveri, D. Kontokostas, S. Hellmann, and J. Umbrich, "Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago," *Semant. Web*, vol. 9, no. 1, p. 77129, 2018.

[22] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *NeurIPS*, vol. 26, 2013.

[23] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, vol. 29, 2015.

[24] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL/IJCNLP*, pp. 687–696, 2015.

[25] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, vol. 28, 2014.

[26] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ACM ICML*, pp. 2071–2080, PMLR, 2016.

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

[28] Y. Zhao and J. Liu, "Scef: A support-confidence-aware embedding framework for knowledge graph refinement," *arXiv*, 2019.

[29] J. Dong, Q. Zhang, X. Huang, Q. Tan, D. Zha, and Z. Zihao, "Active ensemble learning for knowledge graph error detection," in *ACM WSDM*, pp. 877–885, 2023.

[30] Q. Zhang, J. Dong, K. Duan, X. Huang, Y. Liu, and L. Xu, "Contrastive knowledge graph error detection," in *ACM CIKM*, pp. 2590–2599, 2022.

[31] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," in *EMNLP*, pp. 705–714, Sept. 2015.

[32] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *NIPS*, vol. 26, 2013.

[33] A. Arasu, M. Götz, and R. Kaushik, "On active learning of record matching packages," in *ACM SIGMOD*, pp. 783–794, 2010.

[34] A. Arasu, C. Ré, and D. Suciu, "Large-scale deduplication with constraints using dedupalog," in *IEEE ICDE*, pp. 952–963, IEEE, 2009.

[35] X. Chu, I. F. Ilyas, and P. Koutris, "Distributed data deduplication," *VLDB*, vol. 9, no. 11, pp. 864–875, 2016.

[36] W. Fan, F. Geerts, N. Tang, and W. Yu, "Conflict resolution with data currency and consistency," *JDIQ*, vol. 5, no. 1-2, pp. 1–37, 2014.

[37] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas, "Holodetect: Few-shot learning for error detection," in *ACM SIGMOD*, pp. 829–846, 2019.

[38] M. Mahdavi, Z. Abedjan, R. Castro Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Raha: A configuration-free error detection system," in *ACM SIGMOD*, pp. 865–882, 2019.

[39] W. Fan, Y. Wu, and J. Xu, "Functional dependencies for graphs," in *ACM SIGMOD*, pp. 1843–1857, 2016.

[40] W. Fan, X. Liu, P. Lu, and C. Tian, "Catching numeric inconsistencies in graphs," in *ACM TODS*, pp. 381–393, 2018.

[41] W. Fan, R. Jin, M. Liu, P. Lu, C. Tian, and J. Zhou, "Capturing associations in graphs," *VLDB*, vol. 13, no. 12, pp. 1863–1876, 2020.

[42] W. Fan, P. Lu, C. Tian, and J. Zhou, "Deducing certain fixes to graphs," *VLDB*, vol. 12, no. 7, pp. 752–765, 2019.

[43] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," *NIPS*, vol. 31, 2018.

[44] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," *arXiv*, 2019.

[45] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *TNNLS*, vol. 33, no. 2, pp. 494–514, 2021.

[46] B. An, B. Chen, X. Han, and L. Sun, "Accurate text-enhanced knowledge graph representation learning," in *NAACL*, pp. 745–755, June 2018.

[47] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *EMNLP*, pp. 267–272, 2015.

[48] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *AAAI*, vol. 30, 2016.

[49] H. Xiao, M. Huang, L. Meng, and X. Zhu, "Ssp: semantic space projection for knowledge graph embedding with text descriptions," in *AAAI*, vol. 31, 2017.

[50] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013.

[51] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, vol. 32, 2018.

[52] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI*, vol. 24, pp. 1306–1313, 2010.

[53] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *CIDR*, 2013.

[54] G. A. Miller, "Wordnet: a lexical database for english," *CACM*, vol. 38, no. 11, pp. 39–41, 1995.

[55] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[56] Q. Zhang, J. Dong, K. Duan, X. Huang, Y. Liu, and L. Xu, "Contrastive knowledge graph error detection," in *ACM CIKM*, pp. 2590–2599, 2022.

[57] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE ICASSP*, pp. 6645–6649, Ieee, 2013.

[58] C.-P. Lee and C.-J. Lin, "A study on l2-loss (squared hinge-loss) multiclass svm," *Neural computation*, vol. 25, no. 5, pp. 1302–1323, 2013.

APPENDIX

Definition of Semantic Consistency. The path from h to t is $p^u_{h\to t} = (r^u_1, e^u_1, r^u_2, e^u_2, \ldots, e^u_{m-1}, r^u_m)$, and the path from t to h is $p^v_{h\to t} = (r^v_1, e^v_1, r^v_2, e^v_2, \ldots, e^v_{m-1}, r^v_m)$. Then, we define

$$\vec{U} = \vec{r^u_1} + \vec{r^u_2} + \cdots + \vec{r^u_m} = \vec{t} - \vec{h},$$

$$\vec{V} = \vec{r^v_1} + \vec{r^v_2} + \cdots + \vec{r^v_n} = \vec{h} - \vec{t}.$$

Semantic consistency means $||U - V|| \leq \epsilon$, where ϵ is very small. This is the equivalent definition of the definition in Sec.II.

Definition of Semantic Inconsistency. The same as the definition of semantic consistency. Semantic Inconsistency means $||U - V|| \geq \epsilon$. This is also the equivalent definition of the definition in Sec.II.

Proof of Insight 1. A pair of paths $(p^u_{h\to t}, p^v_{t\to h})$, connecting head entity h and tail entity t can be conceptualized as a directed graph. In this graph, nodes correspond to entities, while directed edges represent relations. Consequently, each path can be expressed as an ordered sequence of nodes.

Therefore, the path from h to t can be represented as $E = (e_1, e_2, \ldots, e_g)$, while the path from t to h is $E' = (e'_1, e'_2, \ldots, e'_q)$. Thus, we can represent the pair of paths as follows:

$$< (h, e_1, e_2, \ldots, e_g, t), (t, e'_1, e'_2, \ldots, e'_q, h) > .$$

- If $\forall f \in (e_1, e_2, \ldots, e_g), f \notin (e'_1, e'_2, \ldots, e'_q)$, this situation can be abstracted as illustrated in Fig. 2(a). Consider any node k within the set of nodes N. It is guaranteed that there exists a path from node k to entity t. Moreover, path E' represents the path from entity t to entity h. Therefore, there must exist a path from node k to entity h. Finally, since k is a node in E, it is evident that there exists a path from entity h to node k. Consequently, there must be a path from node k to itself, forming a ring structure. In summary, the presence of a ring structure is confirmed in this case.
- If $\exists f \in (e_1, e_2, \ldots, e_g), f \in (e'_1, e'_2, \ldots, e'_q)$, which can be abstracted as depicted in Fig. 2(b) and Fig. 2(c). In this case, let the set of nodes satisfying these conditions be represented as $F = (f_1, f_2, \ldots, f_j)$, maintaining the order in the path. For E, $F = (f_1, f_2, \ldots, f_j)$, but for E', $F' = (f_j, f_{j-1}, \ldots, f_1)$. Then E and E' can be divided into $(j+1)$ segments by the nodes in F, expressed as

$$E = (e_1, \ldots, f_1, \ldots, f_2, \ldots, f_j, \ldots, e_g).$$

$$E' = (e'_1, \ldots, f_j, \ldots, f_{j-1}, \ldots, f_1, \ldots, e'_q).$$

  - For the segment containing h, there exists a path from h to f_1 within path E, and there is a path from f_1 to h within path E'. As a result, a ring structure is established around the head entity h.
  - For the segment involving t, there exists a path from $f_j \to t$ within path E, and a path from t to f_j within path E'. This forms a ring structure around the tail entity t.
  - For the remaining segments, arbitrarily chosen $f_s \in (f_2, \ldots, f_{j-1})$. There is a path connecting f_s to f_{s+1} within path E, and a path from f_{s+1} to f_s within path E'. Consequently, ring structures are established involving these intermediate nodes.

To sum up, there must be several rings together to form a pair of paths $(p^u_{h\to t}, p^v_{t\to h})$ between h and t.

Proof of Lemma 2. Given an edge $R_{A\to B}$ in a KG, find any two distinct paths from B to A in the KG, represented as $p^1_{B\to A}, p^2_{B\to A}$. The proof for the Lemma from the triple and TransE ideas are provided as follows.

Triple perspective. $R_{A\to B} = Ar_{AB}B$,

$$p^1_{B\to A} = (Br^1_1 n^1_1, n^1_1 r^1_2 n^1_2, \ldots, n^1_i r^1_{i+1} A),$$

$$p^2_{B\to A} = (Br^2_1 n^2_1, n^2_1 r^2_2 n^2_2, \ldots, n^2_j r^2_{j+1} A).$$

Ideally, vector operations are performed. If there is a conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$Br^1_1 n^1_1 + n^1_1 r^1_2 n^1_2 + \cdots + n^1_i r^1_{i+1} A + Ar_{AB}B \neq 0$$

according to the above nature, there is

$$Br^1_1 n^1_1 + n^1_1 r^1_2 n^1_2 + \cdots + n^1_i r^1_{i+1} A =$$

$$Br^2_1 n^2_1 + n^2_1 r^2_2 n^2_2 + \cdots + n^2_i r^2_{i+1} A,$$

and then

$$Br^2_1 n^2_1 + n^2_1 r^2_2 n^2_2 + \cdots + n^2_i r^2_{i+1} A + Ar_{AB}B$$

$$= R_{A\to B} + p^2_{B\to A} \neq 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has conflict.

In the same way, if there is no conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$Br^1_1 n^1_1 + n^1_1 r^1_2 n^1_2 + \cdots + n^1_i r^1_{i+1} A + Ar_{AB}B = 0,$$

according to the above nature, there is

$$Br^1_1 n^1_1 + n^1_1 r^1_2 n^1_2 + \cdots + n^1_i r^1_{i+1} A =$$

$$Br^2_1 n^2_1 + n^2_1 r^2_2 n^2_2 + \cdots + n^2_i r^2_{i+1} A,$$

and then

$$Br^2_1 n^2_1 + n^2_1 r^2_2 n^2_2 + \cdots + n^2_i r^2_{i+1} A + Ar_{AB}B$$

$$= R_{A\to B} + p^2_{B\to A} = 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has no conflict.

TransE *with triple perspective.* In relation perspective without TransE, the paths are $R_{A\to B} = Ar_{AB}B$, $p^1_{B\to A} = (Br^1_1 n^1_1, n^1_1 r^1_2 n^1_2$ and $p^2_{B\to A} = (Br^2_1 n^2_1, n^2_1 r^2_2 n^2_2, \ldots, n^2_j r^2_{j+1} A)$. As the idea of TransE, if $h + r \simeq t$, the paths with relations can be rewrited as $R_{A\to B} = 2B$,

$$p^1_{B\to A} = (2n^1_1, 2n^1_2, \ldots, 2A),$$

$$p^2_{B\to A} = (2n^2_1, 2n^2_2, \ldots, 2A)$$

Ideally, vector operations are performed. If there is a conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$2n^1_1 + 2n^1_2 + \cdots + 2A + 2B \neq 0$$

According to the above nature, there is

$$2n^1_1 + 2n^1_2 + \cdots + 2A =$$

$$2n^2_1 + 2n^2_2 + \cdots + 2A,$$

and then

$$2n^2_1 + 2n^2_2 + \cdots + 2A + 2B$$

$$= R_{A\to B} + p^2_{B\to A} \neq 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has a conflict.

In the same way, if there is no conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$2n^1_1 + 2n^1_2 + \cdots + 2A + 2B = 0,$$

according to the above nature, there is

$$2n^1_1 + 2n^1_2 + \cdots + 2A =$$

$$2n^2_1 + 2n^2_2 + \cdots + 2A,$$

and then

$$2n^2_1 + 2n^2_2 + \cdots + 2A + 2B$$

$$= R_{A\to B} + p^2_{B\to A} = 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has no conflict.

TransE *with relation perspective.* In relation perspective without TransE, the paths are $R_{A\to B} = (r_{AB})$, $p^1_{B\to A} = (r^1_1, r^1_2, \ldots, r^1_{i+1})$ and $p^2_{B\to A} = (r^2_1, r^2_2, \ldots, r^2_{j+1})$. As the idea of TransE, if $h + r \simeq t$, the paths with relations can be rewrited as $R_{A\to B} = (A - B)$

$$p^1_{B\to A} = (B - n^1_1, n^1_1 - n^1_2, \ldots, n^1_i - A),$$

$$p^2_{B\to A} = (B - n^2_1, n^2_1 - n^2_2, \ldots, n^2_j - A)$$

Ideally, vector operations are performed. If there is a conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$(B - n^1_1) + (n^1_1 - n^1_2) + \cdots + (n^1_i - A) + (A - B) \neq 0$$

According to the above nature, there is

$$(B - n^1_1) + (n^1_1 - n^1_2) + \cdots + (n^1_i - A) =$$

$$(B - n^2_1) + (n^2_1 - n^2_2) + \cdots + (n^2_j - A),$$

and then

$$(B - n^2_1) + (n^2_1 - n^2_2) + \cdots + (n^2_j - A) + (A - B)$$

$$= R_{A\to B} + p^2_{B\to A} \neq 0.$$

That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has a conflict.

In the same way, if there is no conflict in the ring structure formed by $R_{A\to B}$ and $p^1_{B\to A}$, then

$$R_{A\to B} + p^1_{B\to A} =$$

$$(B - n^1_1) + (n^1_1 - n^1_2) + \cdots + (n^1_i - A) + (A - B) = 0,$$

according to the above nature, there is

$$Br^1_1 n^1_1 + n^1_1 r^1_2 n^1_2 + \cdots + n^1_i r^1_{i+1} A =$$

$$Br^2_1 n^2_1 + n^2_1 r^2_2 n^2_2 + \cdots + n^2_i r^2_{i+1} A,$$

and then

$$(B - n^2_1) + (n^2_1 - n^2_2) + \cdots + (n^2_j - A) + (A - B)$$

$$= R_{A\to B} + p^2_{B\to A} = 0.$$

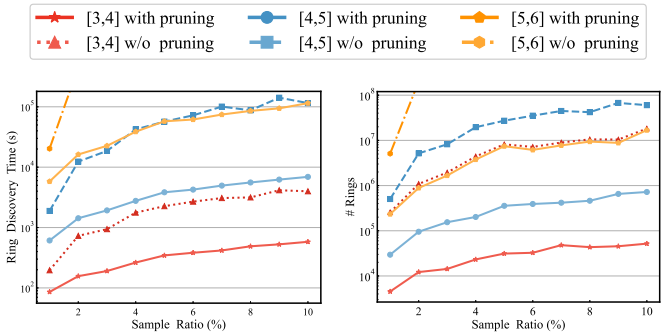
That is to say, the ring structure formed by $R_{A\to B}$ and $p^2_{B\to A}$ also has no conflict.

To sum up, the Lemma 2 holds under both triple and relation perspectives, and holds under both common and TransE perspectives.

Complementary experiment results.

System Efficiency on YAGO *3-10.* We give the detailed experimental results with the sample ratio from 1% to 10% in Fig. 11. The pruning strategy can improve the efficiency of ring discovery around 85% for $[3, 4]$ and around 90%, 95% for $[4, 5]$. Thus, this can also help the conflict detection in static scene up to 85%. And we notice that, the higher the sample ratio, the larger efficiency improvement. So as the number of rings. In Fig. 12(b), we notice that the number of rings fluctuates a bit, while there is an overall increasing trend. That's because, each sample is independent to ensure randomness, and the results of sampling may affect the results.

System Efficiency on NELL-*995.* We also experiment the system efficiency on NELL-995 dataset. NELL-995 also has enought rings for us to sample. We set the sample ratio as 1%, 5%, 10%, 15% and 20%, which are the same as the experiments on YAGO 3-10. And the results are in Fig. 12. Results show that the pruning strategy can improve the efficiency for ring discovery around 60% for $[3, 4]$ and around 70% for $[4, 5]$. The reduction is smaller than YAGO 3-10, because the data sizes of NELL-995 are smaller than that of YAGO 3-10.

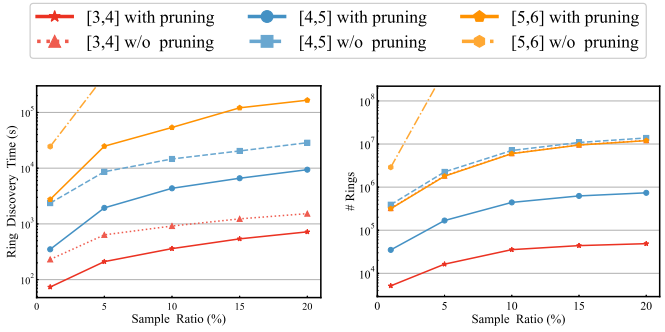


(a) Time consumption. (b) The number of rings.

Fig. 11. Results of Ring Discovery with and without pruning. (a) shows the time consumption with and without pruning. (b) shows the number of rings with and without pruning. The same color with different shades is for the same path length, the dark color represents with pruning, and the light color represents without pruning. The all solid are the results with pruning.

TABLE IV
THE STATISTICS OF SAMPLE RESULT OF NELL-995.

Sample Ratio(%)	# Entities	# Triples
1	14929	26625
5	19566	39165
10	24032	49223
15	27378	55819
20	30689	62774



(a) Time consumption. (b) The number of rings.

Fig. 12. Results of Ring Discovery with and without pruning on NELL-995. (a) shows the time consumption with and without pruning on NELL-995. (b) shows the number of rings with and without pruning on NELL-995. The results out of the figure are those who cannot get the result within 48 hours. The same color with different shades is for the same path length, the dark color represents with pruning, and the light color represents without pruning. The all solid are the results with pruning.