

Logic Rules Powered Knowledge Graph Embedding

Pengwei Wang

Alibaba Group

hoverwang.wpw@alibaba-inc.com

Dejing Dou

University of Oregon

dou@cs.uoregon.edu

Fangzhao Wu

Microsoft

fangzwu@microsoft.com

Nisansa de Silva

University of Oregon

nisansa@cs.uoregon.edu

Lianwen Jin

South China University of

Technology

lianwen.jin@gmail.com

ABSTRACT

Large scale knowledge graph embedding has attracted much attention from both academia and industry in the field of Artificial Intelligence. However, most existing methods concentrate solely on fact triples contained in the given knowledge graph. Inspired by the fact that logic rules can provide a flexible and declarative language for expressing rich background knowledge, it is natural to integrate logic rules into knowledge graph embedding, to transfer human knowledge to entity and relation embedding, and strengthen the learning process. In this paper, we propose a novel logic rule-enhanced method which can be easily integrated with any translation based knowledge graph embedding model, such as TransE [3]. We first introduce a method to automatically mine the logic rules and corresponding confidences from the triples. And then, to put both triples and mined logic rules within the same semantic space, all triples in the knowledge graph are represented as first-order logic. Finally, we define several operations on the first-order logic and minimize a global loss over both of the mined logic rules and the transformed first-order logics. We conduct extensive experiments for link prediction and triple classification on three datasets: WN18, FB166, and FB15K. Experiments show that the rule-enhanced method can significantly improve the performance of several baselines. The highlight of our model is that the filtered Hits@1, which is a pivotal evaluation in the knowledge inference task, has a significant improvement (up to 700% improvement).

KEYWORDS

Knowledge graph embedding, logic rule, rule-enhanced method

ACM Reference format:

Pengwei Wang, Dejing Dou, Fangzhao Wu, Nisansa de Silva, and Lianwen Jin. 2019. Logic Rules Powered Knowledge Graph Embedding. In *Proceedings of, arxiv, paper* (2019), 11 pages.

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

2019, paper, arxiv

© 2019 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Knowledge graphs such as Freebase¹ [1], Wordnet² [15] and YAGO³ [23] play a pivotal role in many Artificial Intelligence related tasks, such as web search, automatic question answering systems, etc. A knowledge graph is a multi-relational data composed of entities as nodes and relations as different types of edges. A fact triple is stored in the form of (head entity, relation, tail entity) (denoted as (h, r, t)), e.g., (Washington, isCapitalof, USA). Although a significant number of large scale knowledge graphs have been constructed, the symbolic nature of such triples makes knowledge graph hard to manipulate, especially in knowledge inference tasks [8].

Recently, a new approach, namely knowledge graph embedding, has been proposed to handle the symbolic nature problems. The goal of the knowledge graph embedding is to embed a knowledge graph into a continuous vector space while preserving certain properties of the original graph [3, 4, 6, 22, 29]. For example, the entity h (or t) is represented as a point \mathbf{h} (or \mathbf{t}) in vector space, and the relation r is represented as a translation operation \mathbf{r} in the same vector space. The embedding representations contain rich semantic information of entities and relations, and global knowledge graph information.

However, most existing methods [3, 27] concentrate solely on fact triples contained in the given knowledge graph but not on other background knowledge. Logic rules provide a flexible and declarative language for expressing structured and rich background knowledge [9]. It is therefore desirable to integrate logic rules into knowledge graph embedding, to transfer human knowledge to entity and relation embedding, and strengthen the learning process. Recently, Wang et al. [25] and Wei et al. [28] attempted to leverage the rules into the knowledge graph embedding. However, the rules are used in the post-processing step, which is separate from the knowledge graph embedding process. Guo et al. [8] first proposed jointly embedding rules and triples. However, they manually selected the rules in the rule mining process, and they did not represent rules and triples in the same space. Although these models are effective in the experiments, there are three limitations in most of the models.

- (1) The logic rules are underused in most knowledge graph embedding models.

¹<https://developers.google.com/freebase/>

²<https://wordnet.princeton.edu/>

³<http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/#c1044>

- (2) Rules need to be manually selected, which leads to the fact that it is hard to extend these methods to a large scale knowledge graph.
- (3) Rule are encoded in the form of true value, which leads to the one-to-many mapping phenomenon. For example, one encoding form can map several rules including correct and incorrect rules.
- (4) Algebraic operations of logic symbols are inconsistent in triples and rules. Thus, it is hard to efficiently jointly embed the rules and the knowledge graph.

In this paper, we propose a novel logic rule-enhanced method which can be easily integrated with any translation based knowledge graph embedding model, such as TransE [3]. Firstly, we introduce a method to automatically mine the logic rules and the corresponding confidences from the triples. In general, rules are composed by several components. In this paper, we consider three types of rules: inference rules, transitivity rules, and antisymmetry rules. Inference rules denote that one relation can imply another relation where the former relation could be a subproperty of the latter relation. Transitivity rules denote the combination of two relations can imply the third relation. Antisymmetry rules denote one relation can imply another relation which has opposite meaning. Only rules with confidences greater than a threshold are used in our rule-enhanced knowledge graph embedding. To make the algebraic operations consistency between triples and rules, we propose to map all of the triples and rules into first-order logics. For example, a triple (h, r, t) can be represented as $r(h) \Rightarrow t$. To guarantee the encoding form of a rule and the rule have one-to-one mapping relation, we propose a general interaction operation for rules to make the components within rules directly interact in the vector space. Finally, we define operations for all of the logical symbols used in the transformed first-order logic. A global loss over the transformed first-order logics is minimized in the knowledge graph embedding process.

We evaluate our rule-enhanced method on link prediction and triple classification on three datasets: WN18, FB166, and FB15K. Experiments show that our proposed rule-enhanced method can significantly improve the performance of several baseline models. Particularly, the filtered Hits@1, which is a pivotal evaluation in the knowledge inference task, has a great improvement (up to 700% improvement). To summarize, the contributions of this paper are as follows:

- (1) We propose a novel logic rule-enhanced method which can be easily integrated with any translation based knowledge graph embedding model.
- (2) We introduce a method to automatically mine logic rules and corresponding confidences from the triples in a given knowledge graph.
- (3) We propose to transform the triples and logic rules into the same first-order logic space and encode the rules in the vector space.
- (4) Experiments show that our method brings a great improvement in the filtered Hits@1 evaluation.

2 PROBLEM DEFINITION

In this section, we first give a formal definition of the knowledge graph embedding. A knowledge graph \mathcal{G} is defined as a set of triples $\mathcal{K} = (h, r, t)$, with each triple composed of two entities $h, t \in \mathcal{E}$ and their corresponding relation $r \in \mathcal{R}$. \mathcal{E} denotes the entity vocabulary in \mathcal{G} and \mathcal{R} denotes the relation vocabulary in \mathcal{G} . h denotes the head entity of a triple and t denotes the tail entity of a triple. The goal of knowledge graph embedding is to learn the embedding representations of entities and relations for easily computing in continuous space, e.g., computing the similarity of two entities. We use bold letters $\mathbf{h}, \mathbf{r}, \mathbf{t}$ to denote the embedding of h, r, t , where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$.

3 RULE EXTRACTION

We introduce three types of rules, which are used in our proposed model.

- **Rule 1 (inference rule).** An inference rule is in the form of $\forall h, t : (h, r_1, t) \Rightarrow (h, r_2, t)$. The relation r_1 can imply relation r_2 , which denotes that any two entities linked by r_1 should also be linked by r_2 . For example, (Washington, isCapitalof, USA) \Rightarrow (Washington, isLocatedin, USA). An inference rule is directed. We define the relation r_2 to be the concept of the relation r_1 (r_1 is an instance of r_2).
- **Rule 2 (transitivity rule).** A transitivity rule is in the form of $\forall e_1, e_2, e_3 : [(e_1, r_1, e_2) + (e_2, r_2, e_3) \Rightarrow (e_1, r_3, e_3)]$. The orderly conjunction of relation r_1 and relation r_2 can imply relation r_3 . The transitivity rule denotes that if e_1 and e_2 are linked by relation r_1 and e_2 and e_3 are linked by relation r_2 , e_1 and e_3 will be linked by relation r_3 . For example, (George W. Bush, Nationality, USA) + (USA, Official_Language, English) \Rightarrow (George W. Bush, Language, English).
- **Rule 3 (antisymmetry rule).** An antisymmetry rule is in the form of $\forall h, t : (h, r_1, t) \Leftrightarrow (t, r_2, h)$. The relation r_1 can imply the antisymmetry relation r_2 , which denotes that two relations r_1 and r_2 are antisymmetrical. For example, (apple, hypernym, fruit) \Leftrightarrow (fruit, hyponym, apple). An antisymmetry rule is undirected.

As shown in Figure 1, the input of this framework is triples of the knowledge graph \mathcal{G} , and the output is the ground rules of the three types with corresponding scores. There are three steps in this framework.

3.1 Rule Sample Extraction

In this step, we extract the rule samples from given triples. Here, we define the rule samples as the triple combinations that meet the conditions of rules. For example, a rule 1 sample composed the ID 1 triple and ID 2 triple in the "Rule 1 Samples" table of Figure 1 can meet the inference rule because the two triples have the same head entity and tail entity; a rule 2 sample composed of the ID 1 triple, ID 3 triple and ID 4 triple in the "Rule 2 Samples" table of Figure 1 can meet the transitivity rule because of the sequential entities (EID_1, EID_2 , and EID_3); a rule 3 sample composed of the ID 3 triple and ID 5 triple in the "Rule 3 Samples" table of Figure 1 can meet the antisymmetry rule because the two triples have

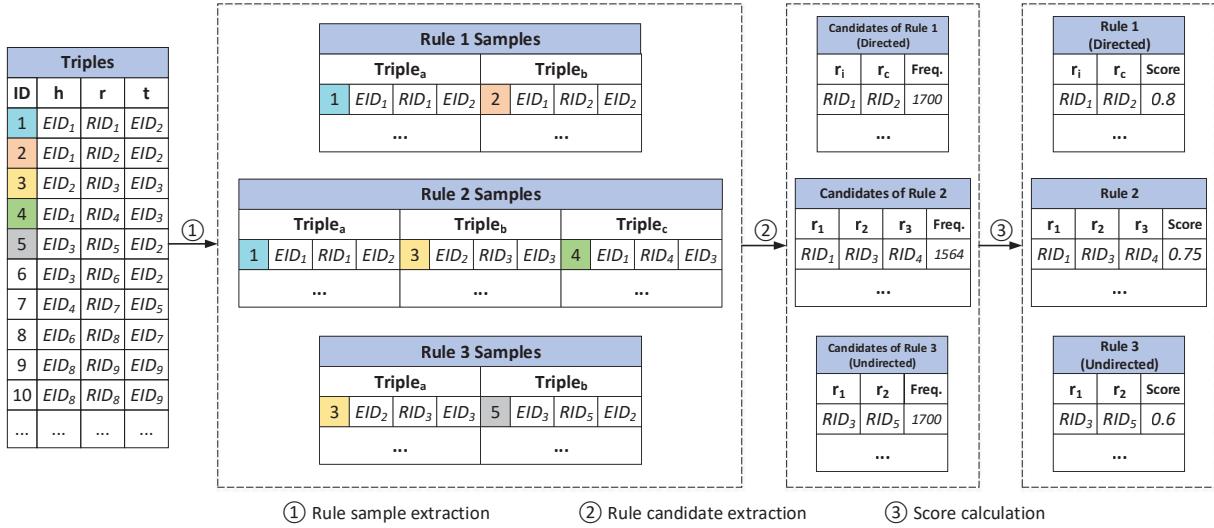


Figure 1: The flowchart of rule extraction.

the antisymmetric head entity and tail entity. Many rule samples belonging to the above three types of rules are mined using this manner.

3.2 Rule Candidate Extraction

We extract some candidates for rules and do some statistics according to the rule samples. For example, (RID₁ and RID₂) in the "Candidates of Rule 1" table of Figure 1 are extracted from each sample of rule 1; (RID₁, RID₃, and RID₄) in the "Candidates of Rule 2" table of Figure 1 are extracted from each sample of rule 2; (RID₃ and RID₅) in the "Candidates of Rule 3" table of Figure 1 are extracted from each sample of rule 3. All candidates of three types of rules are extracted from the rule samples using this way. Then, we rank the candidates in descending order according to the frequencies. Because the inference rule is directed, we need to distinguish the concept relation and the instance relation. The relations in Freebase are hierarchical. For example, the /location/country/capital relation has three diminishing levels (location, country, and capital). To judge the concept relation between /location/country and /location/location/contains, we only judge the concept from the second level, namely country and location, since the first levels are the same. In Freebase, each entity has several object types, as shown in Table 1. The object type is also hierarchical. We can get the concept level from the object type. For example, we obtain that location is the concept of country from the object type location.country. Thus, the relation /location/location/contains is the concept of the relation /location/country/capital. In addition, we can also get the concept of an entity by leveraging Probase⁴ [30]. Given a word, Probase will provide the concepts associated with the word. More usages of Probase can be found in its official website.

Table 1: Entity object types.

Relation	Concept-Instance Pairs
location.country	(location-country)
people.profession	(people-profession)
music.songwriter	(music-songwriter)
sports.boxer	(sports-boxer)
book.magazine	(book-magazine)

3.3 Score Calculation

Many candidate rules are not correct. Not all results inferred by the candidate rules are reasonable. In KALE [8], the wrong rules ranking at the top are manually filtered. It is hard to leverage KALE in a large scale knowledge graph, which has a large number of relations. We propose a novel method to automatically select the rules from the candidate pool. Algorithm 1 summarizes the proposed method. We first generate all of the new triples according to each candidate rule using the function *GetNewtriples()*, which is introduced as follows.

- Inference candidate rule: Given a candidate rule $r_1 \Rightarrow r_2$, if a triple has the relation r_1 , such as (h, r_1, t) , a new triple (h, r_2, t) can be generated. Each candidate can generate triples in the same manner.
- Transitivity candidate rule: Given a candidate rule $(r_1 + r_2) \Rightarrow r_3$, if there are two triples (e_1, r_1, e_2) and (e_2, r_2, e_3) , a new triple (e_1, r_3, e_3) can be generated. Each candidate can generate triples in the same manner.
- Antisymmetry candidate rule: Given a candidate rule $r_1 \Leftrightarrow r_2$, if a triple has the relation r_1 , such as (h, r_1, t) , a new triple (t, r_2, h) can be generated. Each candidate can generate triples in the same manner.

As shown in Algorithm 1, we can calculate the percentage α_i about the new triples that exist in original triples \mathcal{K} for each candidate rule. We only select the rules whose percentages are greater

⁴<https://concept.msra.cn/Home/Download>

Algorithm 1 Score Calculation

Input: Candidate rules R_a , triples \mathcal{K} , threshold τ
Output: Ultimate rules R_u

- 1: $T \leftarrow \emptyset$
- 2: **for** each rule r in R_a **do**
- 3: $T_n \leftarrow \text{GetNewtriples}(r)$
- 4: $T \leftarrow T \cup T_n$
- 5: **end for**
- 6: $A \leftarrow \emptyset$
- 7: **for** each new triple set T_n in T **do**
- 8: $\alpha_i = \frac{\#(T_n \cup \mathcal{K})}{\#T_n}$
- 9: $A \leftarrow A \cup \alpha_i$
- 10: **end for**
- 11: **for** $i = 1$ to $\#(R_a)$ **do**
- 12: **if** $A[i] \geq \tau$ **then**
- 13: $R_u \leftarrow R_u \cup R_a[i]$
- 14: **end if**
- 15: **end for**

than the threshold τ . Then, we generate the ground rules for each rule in the triples \mathcal{K} . For example, given a rule $r_1 \Rightarrow r_2$, we can generate a corresponding ground rule $(h_m, r_1, t_n) \Rightarrow (h_m, r_2, t_n)$ from a triple (h_m, r_1, t_n) . Finally, we obtain all of the ground rules for each rule.

4 RULE-ENHANCED KNOWLEDGE GRAPH EMBEDDING

The proposed rule-enhanced knowledge graph embedding method can be easily used in any translation based knowledge graph embedding model, such as TransE [3]. In this paper, we apply it into not only TransE [3], but also TransH [27] and TransR [13].

4.1 Rule-Enhanced TransE Model

We follow TransE [3] to model the triples. TransE represents a relation by a translation vector \mathbf{r} so that the pair of the two embedded entities in a triple (h, r, t) can be linked by \mathbf{r} . The score function $s_1(h, r, t)$ of TransE is:

$$s_1(h, r, t) = \| \mathbf{h} + \mathbf{r} - \mathbf{t} \|_{l_{1/2}}, \quad (1)$$

where \mathbf{h} and \mathbf{t} are the learned entity embeddings. \mathbf{r} is the learned relation embedding, and $l_{1/2}$ denotes the L_1 -norm or L_2 -norm respectively.

Then, we try to model the ground rules into the knowledge graph embedding process. As shown in Table 2, we formulate the knowledge graph and the three types of ground rules as first-order logic. In the reference ground rule (the second row in Table 2), C denotes the concept of entity h in the triple (h, r_2, t) , which can be obtained in Section 3. The head entity h in two relations r_1 and r_2 has different concepts. For example, in this inference ground rule (Washington, isCapitalof, USA) \Rightarrow (Washington, isLocatedin, USA), the concepts of the first and second Washington are capital and location respectively. In a inference ground rule, we assume the head entity must be an instance of the concept of the head in second relation if the first relation r_1 want to deduce the second relation r_2 . In addition, we assume that the entity h can deduce the entity t after that the entity h is translated by relation r .

After formulating the knowledge graph and the three types of ground rules as first-order logic, some logical symbols, such as $r(\cdot)$, \Rightarrow , \in , and \wedge , are defined in Table 3. The \otimes denotes pairwise multiplication. The a and b can be an entity, a triple or a combination of triples.

We consider three types of ground rules. The first type is $\forall h, t : (h, r_1, t) \Rightarrow (h, r_2, t)$. Given a ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (h_m, r_2, t_n)$, the score function is calculated by:

$$s_2(f) = \| (\mathbf{h}_m \cdot C) \otimes (\mathbf{h}_m + \mathbf{r}_1 - \mathbf{t}_n) - (\mathbf{h}_m + \mathbf{r}_2 - \mathbf{t}_n) \|_{l_{1/2}}. \quad (2)$$

The second type is $\forall e_1, e_2, e_3 : [(e_1, r_1, e_2) \wedge (e_2, r_2, e_3) \Rightarrow (e_1, r_3, e_3)]$. Given a ground rule $f \triangleq [(e_l, r_1, e_m) \wedge (e_m, r_2, e_n) \Rightarrow (e_l, r_3, e_n)]$, the score function is calculated by:

$$\begin{aligned} s_3(f) = & \| [(\mathbf{e}_l + \mathbf{r}_1 - \mathbf{e}_m) \otimes (\mathbf{e}_m + \mathbf{r}_2 - \mathbf{e}_n)] \\ & - (\mathbf{e}_l + \mathbf{r}_3 - \mathbf{e}_n) \|_{l_{1/2}}. \end{aligned} \quad (3)$$

The third type is $\forall h, t : (h, r_1, t) \Rightarrow (t, r_2, h)$. Given a ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (t_n, r_2, h_m)$, the score function is calculated by:

$$s_4(f) = \| (\mathbf{TR}_f - \mathbf{TR}_b) \otimes (\mathbf{TR}_b - \mathbf{TR}_f) \|_{l_{1/2}}, \quad (4)$$

$$\mathbf{TR}_f = \mathbf{h}_m + \mathbf{r}_1 - \mathbf{t}_n, \quad \mathbf{TR}_b = \mathbf{t}_n + \mathbf{r}_2 - \mathbf{h}_m. \quad (5)$$

4.2 Rule-Enhanced TransH Model

To address the issue of TransE when modeling N-to-1, 1-to-N and N-to-N relations, Wang et al. [27] proposed TransH to enable an entity to have distinct embeddings when involved in different relations. For a relation r , TransH models the relations as a vector \mathbf{r} on a hyperplane with \mathbf{w}_r as the normal vector. The score function $s_1(h, r, t)$ of TransH is:

$$s_1(h, r, t) = \| (\mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r) + \mathbf{r} - (\mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r) \|_{l_{1/2}}, \quad (6)$$

where \mathbf{h}, \mathbf{t} are the learned entity embeddings, \mathbf{r} is the learned relation embedding and \mathbf{w}_r is the learned normal vector.

As introduced in section 4.1, we apply the three types of ground rules in the TransH model. For the inference ground rule, given a ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (h_m, r_2, t_n)$, the score function is calculated by:

$$s_2(f) = \| (\mathbf{h}_m \cdot C) \otimes (\mathbf{h}_{\perp 1}^m + \mathbf{r}_1 - \mathbf{t}_{\perp 1}^n) - (\mathbf{h}_{\perp 2}^m + \mathbf{r}_2 - \mathbf{t}_{\perp 2}^n) \|_{l_{1/2}}, \quad (7)$$

$$\mathbf{h}_{\perp 1}^m = \mathbf{h}_m - \mathbf{w}_{r_1}^\top \mathbf{h}_m \mathbf{w}_{r_1}, \quad \mathbf{t}_{\perp 1}^n = \mathbf{t}_n - \mathbf{w}_{r_1}^\top \mathbf{t}_n \mathbf{w}_{r_1}, \quad (8)$$

$$\mathbf{h}_{\perp 2}^m = \mathbf{h}_m - \mathbf{w}_{r_2}^\top \mathbf{h}_m \mathbf{w}_{r_2}, \quad \mathbf{t}_{\perp 2}^n = \mathbf{t}_n - \mathbf{w}_{r_2}^\top \mathbf{t}_n \mathbf{w}_{r_2}. \quad (9)$$

Given a transitivity ground rule $f \triangleq [(e_l, r_1, e_m) \wedge (e_m, r_2, e_n) \Rightarrow (e_l, r_3, e_n)]$, the score function is calculated by:

$$\begin{aligned} s_3(f) = & \| [(\mathbf{e}_{\perp 1}^l + \mathbf{r}_1 - \mathbf{e}_{\perp 1}^m) \otimes (\mathbf{e}_{\perp 2}^m + \mathbf{r}_2 - \mathbf{e}_{\perp 2}^n)] \\ & - (\mathbf{e}_{\perp 3}^l + \mathbf{r}_3 - \mathbf{e}_{\perp 3}^n) \|_{l_{1/2}}, \end{aligned} \quad (10)$$

$$\mathbf{e}_{\perp 1}^l = \mathbf{e}_l - \mathbf{w}_{r_1}^\top \mathbf{e}_l \mathbf{w}_{r_1}, \quad \mathbf{e}_{\perp 1}^m = \mathbf{e}_m - \mathbf{w}_{r_1}^\top \mathbf{e}_m \mathbf{w}_{r_1}, \quad (11)$$

$$\mathbf{e}_{\perp 2}^m = \mathbf{e}_m - \mathbf{w}_{r_2}^\top \mathbf{e}_m \mathbf{w}_{r_2}, \quad \mathbf{e}_{\perp 2}^n = \mathbf{e}_n - \mathbf{w}_{r_2}^\top \mathbf{e}_n \mathbf{w}_{r_2}, \quad (12)$$

$$\mathbf{e}_{\perp 3}^l = \mathbf{e}_l - \mathbf{w}_{r_3}^\top \mathbf{e}_l \mathbf{w}_{r_3}, \quad \mathbf{e}_{\perp 3}^n = \mathbf{e}_n - \mathbf{w}_{r_3}^\top \mathbf{e}_n \mathbf{w}_{r_3}. \quad (13)$$

Given an antisymmetry ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (t_n, r_2, h_m)$, the score function is calculated by:

$$s_4(f) = \| (\mathbf{TR}_f - \mathbf{TR}_b) \otimes (\mathbf{TR}_b - \mathbf{TR}_f) \|_{l_{1/2}}, \quad (14)$$

$$\mathbf{TR}_f = \mathbf{h}_{\perp 1}^m + \mathbf{r}_1 - \mathbf{t}_{\perp 1}^n, \quad \mathbf{TR}_b = \mathbf{t}_{\perp 2}^n + \mathbf{r}_2 - \mathbf{h}_{\perp 2}^m. \quad (15)$$

$$\mathbf{h}_{\perp 1}^m = \mathbf{h}_m - \mathbf{w}_{r_1}^\top \mathbf{h}_m \mathbf{w}_{r_1}, \quad \mathbf{t}_{\perp 1}^n = \mathbf{t}_n - \mathbf{w}_{r_1}^\top \mathbf{t}_n \mathbf{w}_{r_1}, \quad (16)$$

Table 2: The form of first-order logic.

Triple and ground rule	The format of first-order logic.
(h, r, t)	$r(h) \Rightarrow t$
$(h, r_1, t) \Rightarrow (h, r_2, t)$	$[(h \in C) \wedge [r_1(h) \Rightarrow t]] \Rightarrow [r_2(h) \Rightarrow t]$
$(e_1, r_1, e_2) + (e_2, r_2, e_3) \Rightarrow (e_1, r_3, e_3)$	$[[r_1(e_1) \Rightarrow e_2] \wedge [r_2(e_2) \Rightarrow e_3]] \Rightarrow [r_3(e_1) \Rightarrow e_3]$
$(h, r_1, t) \Leftrightarrow (t, r_2, h)$	$[[r_1(h) \Rightarrow t]] \Rightarrow [r_2(t) \Rightarrow h] \wedge [[r_2(t) \Rightarrow h]] \Rightarrow [r_1(h) \Rightarrow t]$

Table 3: Mathematical expression of first-order logic.

First-order logic	Mathematical expression
$r(h)$	$\mathbf{r} + \mathbf{h}$
$a \Rightarrow b$	$\mathbf{a} - \mathbf{b}$
$h \in C$	$\mathbf{h} \cdot \mathbf{C}$ (\mathbf{C} is a matrix)
$a \wedge b$	$\mathbf{a} \otimes \mathbf{b}$
$a \Leftrightarrow b$	$(\mathbf{a} - \mathbf{b}) \otimes (\mathbf{a} - \mathbf{b})$

$$\mathbf{h}_{\perp 2}^m = \mathbf{h}_m - \mathbf{w}_{r_2}^\top \mathbf{h}_m \mathbf{w}_{r_2}, \quad \mathbf{t}_{\perp 2}^n = \mathbf{t}_n - \mathbf{w}_{r_2}^\top \mathbf{t}_n \mathbf{w}_{r_2}. \quad (17)$$

4.3 Rule-Enhanced TransR Model

The entities in TransR [13] are mapped into vectors in different relation space embedding according to a relation. Thus, the score function $s_1(h, r, t)$ of TransR is:

$$s_1(h, r, t) = \| \mathbf{hM}_r + \mathbf{r} - \mathbf{tM}_r \|_{l_{1/2}}, \quad (18)$$

where \mathbf{h}, \mathbf{t} are the learned entity embeddings, \mathbf{r} is the learned relation embedding and \mathbf{M}_r is the learned projection matrix.

As introduced in section 4.1, we apply the three types of ground rules in the TransH model. For the inference ground rule, given a ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (h_m, r_2, t_n)$, the score function is calculated by:

$$s_2(f) = \| (\mathbf{h}_m \cdot \mathbf{C}) \otimes (\mathbf{h}_{r_1}^m + \mathbf{r}_1 - \mathbf{t}_{r_1}^n) - (\mathbf{h}_{r_2}^m + \mathbf{r}_2 - \mathbf{t}_{r_2}^n) \|_{l_{1/2}}, \quad (19)$$

$$\mathbf{h}_{r_1}^m = \mathbf{h}_m \mathbf{M}_{r_1}, \quad \mathbf{t}_{r_1}^n = \mathbf{t}_n \mathbf{M}_{r_1}, \quad (20)$$

$$\mathbf{h}_{r_2}^m = \mathbf{h}_m \mathbf{M}_{r_2}, \quad \mathbf{t}_{r_2}^n = \mathbf{t}_n \mathbf{M}_{r_2}. \quad (21)$$

Given a transitivity ground rule $f \triangleq [(e_l, r_1, e_m) + (e_m, r_2, e_n) \Rightarrow (e_l, r_3, e_n)]$, the score function is calculated by:

$$s_3(f) = \| [(\mathbf{e}_{r_1}^l + \mathbf{r}_1 - \mathbf{e}_{r_1}^m) \otimes (\mathbf{e}_{r_2}^m + \mathbf{r}_2 - \mathbf{e}_{r_2}^n) - (\mathbf{e}_{r_3}^l + \mathbf{r}_3 - \mathbf{e}_{r_3}^n)] \|_{l_{1/2}}, \quad (22)$$

$$\mathbf{e}_{r_1}^l = \mathbf{e}_l \mathbf{M}_{r_1}, \quad \mathbf{e}_{r_1}^m = \mathbf{e}_m \mathbf{M}_{r_1}, \quad (23)$$

$$\mathbf{e}_{r_2}^m = \mathbf{e}_m \mathbf{M}_{r_2}, \quad \mathbf{e}_{r_2}^n = \mathbf{e}_n \mathbf{M}_{r_2}, \quad (24)$$

$$\mathbf{e}_{r_3}^l = \mathbf{e}_l \mathbf{M}_{r_3}, \quad \mathbf{e}_{r_3}^n = \mathbf{e}_n \mathbf{M}_{r_3}. \quad (25)$$

Given an antisymmetry ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (t_n, r_2, h_m)$, the score function is calculated by:

$$s_4(f) = \| (\mathbf{TR}_f - \mathbf{TR}_b) \otimes (\mathbf{TR}_b - \mathbf{TR}_f) \|_{l_{1/2}}, \quad (26)$$

$$\mathbf{TR}_f = \mathbf{h}_{r_1}^m + \mathbf{r}_1 - \mathbf{t}_{r_1}^n, \quad \mathbf{TR}_b = \mathbf{t}_{r_2}^n + \mathbf{r}_2 - \mathbf{h}_{r_2}^m. \quad (27)$$

$$\mathbf{h}_{r_1}^m = \mathbf{h}_m \mathbf{M}_{r_1}, \quad \mathbf{t}_{r_1}^n = \mathbf{t}_n \mathbf{M}_{r_1}, \quad (28)$$

$$\mathbf{h}_{r_2}^m = \mathbf{h}_m \mathbf{M}_{r_2}, \quad \mathbf{t}_{r_2}^n = \mathbf{t}_n \mathbf{M}_{r_2}. \quad (29)$$

Table 4: Gate state.

Sample	Gate
$f = \text{triple}$	$I_1(f) = 1, I_2(f) = I_3(f) = I_4(f) = 0,$
$f = \text{ground rule 1}$	$I_2(f) = 1, I_1(f) = I_3(f) = I_4(f) = 0,$
$f = \text{ground rule 2}$	$I_3(f) = 1, I_1(f) = I_2(f) = I_4(f) = 0,$
$f = \text{ground rule 3}$	$I_4(f) = 1, I_1(f) = I_2(f) = I_3(f) = 0,$

5 GLOBAL OBJECTIVE FUNCTION

We minimize a global loss over triples and ground rules to learn the entity and relation embedding representation. Our training dataset \mathcal{F} contains the original triples and the three types of ground rules. Then, we minimize the following loss function to learn the entity embedding and relation embedding from ground rules and original triples:

$$\min_{\mathbf{e}, \mathbf{r}} \sum_{f^+ \in \mathcal{F}} \sum_{f^- \in \mathcal{N}_{f^+}} [S]_+, \quad (30)$$

$$\begin{aligned} S = & I_1(f^+) \cdot (\gamma + s_1(f^+) - s_1(f^-)) \\ & + I_2(f^+) \cdot (\gamma + s_2(f^+) - s_2(f^-)) \\ & + I_3(f^+) \cdot (\gamma + s_3(f^+) - s_3(f^-)) \\ & + I_4(f^+) \cdot (\gamma + s_4(f^+) - s_4(f^-)), \end{aligned} \quad (31)$$

where $s_1()$, $s_2()$, $s_3()$, and $s_4()$ are introduced in Section 4. $I_n(f)$ denotes the gate, which is shown in Table 4. When a sample is selected, only one gate is active each time. f^+ is a positive sample, such as a triple or a ground rule, f^- is a negative sample constructed by corrupting f^+ , $\gamma > 0$ is a margin separating positive and negative samples, and $[x]_+ \triangleq \max\{0, x\}$ denotes the positive part of x . The following constraints are considered when we minimize the loss function:

$$\forall \mathbf{e} \in \mathcal{E}, \|\mathbf{e}\|_2 \leq 1, \quad \forall \mathbf{r} \in \mathcal{R}, \|\mathbf{r}\|_2 \leq 1. \quad (32)$$

If f^+ is a triple sample (h, r, t) , the f^- is composed of the triple with either the head or tail replaced by a random entity (but not both at the same time), which is shown as the following equation:

$$\mathcal{N}_{f^+} = \{(h', r, t' | h' \in \mathcal{E}) \cup (h, r, t' | t' \in \mathcal{E})\}. \quad (33)$$

If f^+ is a ground rule sample of rule 1 $(h, r_1, t) \Rightarrow (h, r_2, t)$, the f^- can be also constructed by replacing the head or tail with a random entity according to Equation (34):

$$\begin{aligned} \mathcal{N}_{f^+} = & \{[(h', r_1, t) \Rightarrow (h', r_2, t) | h' \in \mathcal{E}] \\ & \cup [(h, r_1, t') \Rightarrow (h, r_2, t') | t' \in \mathcal{E}]\}. \end{aligned} \quad (34)$$

If f^+ is a ground rule sample of rule 2 $(e_1, r_1, e_2) + (e_2, r_2, e_3) \Rightarrow (e_1, r_3, e_3)$, the f^- can be constructed by replacing the e_1 or e_3

with a random entity according to Equation (35):

$$\begin{aligned} \mathcal{N}_{f^+} = & \{(e'_1, r_1, e_2) + (e_2, r_2, e_3) \Rightarrow (e'_1, r_3, e_3) | e'_1 \in \mathcal{E}\} \\ & \cup [(e_1, r_1, e_2) + (e_2, r_2, e'_3) \Rightarrow (e_1, r_3, e'_3) | e'_3 \in \mathcal{E}]\}. \end{aligned} \quad (35)$$

If f^+ is a ground rule sample of rule 3 $(h, r_1, t) \Leftrightarrow (t, r_2, h)$, the f^- can be also constructed by replacing the head or tail with a random entity according to Equation (36):

$$\begin{aligned} \mathcal{N}_{f^+} = & \{((h', r_1, t) \Leftrightarrow (t, r_2, h') | h' \in \mathcal{E} \} \\ & \cup [(h, r_1, t') \Leftrightarrow (t', r_2, h) | t' \in \mathcal{E}]\}. \end{aligned} \quad (36)$$

All of the corrupted triples or ground rules (f^-) cannot exist in the original triples and ground rules. The loss function, shown in Equation (30), favors lower values of the energy for training triples or rules than for corrupted triples or rules, and this is a natural implementation of the intended criterion. In this paper, the embedding representation of an entity is the same when the entity appears in the head position and tail position of a triple. The optimization is carried out by stochastic gradient descent, over the possible \mathbf{h} , \mathbf{r} and \mathbf{t} .

6 DISCUSSIONS

we discuss several practical issues of our proposed model. **Uniqueness.** KALE [8] encode the rules in the form of true value. For example, KALE encode an inference ground rule $f \triangleq (h_m, r_1, t_n) \Rightarrow (h_m, r_2, t_n)$ using the following Equation:

$$I(f) = s(h_m, r_1, t_n) \cdot s(h_m, r_2, t_n) - s(h_m, r_1, t_n) + 1. \quad (37)$$

where $s()$ can be Equation 1, 6, or 18. However, this rule modeling method only consider the true value of a triple. Many different triples can generate a same score using $s()$. Thus, many incorrect rules can easily meet the Equation 37. The knowledge graph embedding model would inference inaccurate results according this rule modeling method after learning. In contrast, to guarantee the encoding form of a rule and the rule have one-to-one mapping relation, we propose a general interaction operation for rules to make the components within rules directly interact in the vector space. As shown in Section 4.1, Section 4.2, and Section 4.3, each component interact with each other by the algebraic operation in the vector space, and then our rule modeling method return a true value.

Consistency. We propose to map all of the triples and rules into first-order logics to achieve the algebraic operations consistency. For a triple (h, r, t) , the relation r is regarded as a function operating on the head entity h and the tail entity t can be deduced after function operation in first-order logic. Thus, a triple (h, r, t) can be represented as $r(h) \Rightarrow t$. Similarly, we can also obtain the first-order logic forms of the ground rules as shown in Table 2. The common logic symbols between triples and rules are \Rightarrow and \wedge , whose algebraic operations have been defined in Table 3. Thus, the triples and rules use the same set of algebraic operations. In contrast, KALE [8] use the Equation 37 to express the \Rightarrow symbol in rule model and use subtraction operation in triple.

7 EXPERIMENTS

In this section, we introduce the experimental design, datasets, and results of our proposed model. Through these experiments, we

Table 5: Statistic of datasets.

Dataset	#E	#R	#Trip.(Train / Valid / Test)
FB15K	14,951	1,345	483,142 50,000 59,071
FB166	9,658	166	100,289 10,457 12,327
WN18	40,943	18	141,442 5,000 5,000

want to verify whether our rule-enhanced method can improve the performance of the translation based knowledge graph embedding models. We use a library for parallel knowledge graph embedding, ParaGraphE⁵ [18], to run several baseline models and implement our model. The results of baselines may be slightly different from the published results due to the asynchronous update in multi-thread setting.

7.1 Datasets

Wordnet [15]: This knowledge base is designed to generate an intuitively usable combination of dictionary and thesaurus, and it is used in natural language processing tasks. Its entities (termed synsets) correspond to word senses, and relationships define lexical relations between them. We used the same data version used in [2–4]: WN18, a subset of Wordnet. Examples of triples are $(\text{score_NN_1}, \text{hypernym}, \text{evaluation_NN_1})$ or $(\text{score_NN_2}, \text{has_part}, \text{musical_notation})$.

Freebase [1]: Freebase is a large collaborative and growing knowledge base of general facts. We use the same data version used in [3, 4]: FB15K, a relatively dense subgraph of Freebase where all entities are present in Wikilinks database. In addition, we follow KALE [8] to create a small data version from FB15K with a little difference from KALE. KALE keeps the triples whose relation types start from “people”, “location”, or “sports” in FB15K. After that, there are totally 192 such relations. Guo et al. [8] manually filtered some ambiguous relations, which is not introduced in detail. To approximate the dataset generated by Guo et al. [8], we only keep relations whose object type of head entity and tail entity also start from “people”, “location”, and “sports” from above 192 relations. Finally, there are only 166 relations. This dataset is denoted as FB166 in the rest of this section.

The WN18 and FB15K are released in [3]. For WN18 and FB15K we use the original data split, and for FB166 we extract triples associated with the 166 relations from the training, validation, and testing sets of FB15K. Table 5 gives more details of the three datasets, including the number of entities, relations, triples in train/valid/test set.

We further construct the three types of logic rules for each dataset using the method described in Section 3. Table 6 shows the details of the mined rules: rule 1 (inference rule), rule 2 (transitivity rule), and rule 3 (antisymmetry rule). For the FB166 and FB15K dataset, the thresholds τ of the rule 1, rule 2 and rule 3 are set as 0.5, 0.6, and 0.5 respectively. For the WN18 dataset, the thresholds τ of the rule 1, rule 2 and rule 3 are set as 0.5, 0.5, 0.5 respectively. All of the thresholds are selected according to the validation set. After mining the logical rules, the mined rules are then instantiated with

⁵<https://github.com/LIBBLE/LIBBLE-MultiThread/tree/master/ParaGraphE>

Table 6: The rule statistic.

Dataset	#Rule 1	#Rule 2	#Rule 3	Total
FB15K	160	2537	637	3334
FB166	38	157	73	268
WN18	0	0	11	11

Table 7: The ground rule statistic

Dataset	#Rule 1	#Rule 2	#Rule 3	Total
FB15K	8,508	152,987	113,476	274,971
FB166	2,254	20,444	20,946	43,644
WN18	0	0	9,352	9,352

concrete entities (grounding) using the method described in Section 3. The statistics of ground rules are shown in Table 7. Table 8 shows some examples of the rules in three data sets.

7.2 Link Prediction

This task is designed to complete a triple (h, r, t) with h or t missing. Instead of requiring the best answer, this task focuses on the ranking of candidate entities from the knowledge graph.

Evaluation protocol: We use the same metric as TransE [3]: for each testing triple (h, r, t) , we replace the head entity h by every entity e in the knowledge graph and calculate a dissimilarity score (by the Equation (1), (6), or (18)) on the corrupted triple (e, r, t) . The rank of the original correct triple is obtained after ranking the scores in ascending order. Similarly, we can get another rank for (h, r, t) by corrupting the tail t . Aggregating over all the testing triples \mathcal{K}_t , we use three metrics to do the evaluation:

- (1) **MR:** the value of averaged rank or Mean Rank. The smaller the better. MR is calculated by:

$$MR = \frac{1}{2\#\mathcal{K}_t} \sum_{i=1}^{\#\mathcal{K}_t} (rank_{ih} + rank_{it}), \quad (38)$$

where $rank_{ih}$ and $rank_{it}$ refers to the rank position of the head correct triple for the i^{th} triple by corrupting the head and tail entity.

- (2) **MRR:** the value of mean reciprocal rank. The higher, the better. The reciprocal rank is the multiplicative inverse of the rank of the correct triple. The mean reciprocal rank is the average of the reciprocal ranks of results for all triples. MRR is calculated by:

$$MRR = \frac{1}{2\#\mathcal{K}_t} \sum_{i=1}^{\#\mathcal{K}_t} \left(\frac{1}{rank_{ih}} + \frac{1}{rank_{it}} \right), \quad (39)$$

where $rank_{ih}$ and $rank_{it}$ refers to the same meaning as introducing in MR metric.

- (3) **Hits@n:** the proportion of rank not larger than n . The higher, the better. Hits@10, Hits@5, Hits@3, and Hits@1

are used in our experiments. Hits@n is calculated by:

$$Hits@n = \frac{1}{2\#\mathcal{K}_t} \sum_{i=1}^{\#\mathcal{K}_t} (I_n(rank_{ih}) + I_n(rank_{it})), \quad (40)$$

$$I_n(rank_i) = \begin{cases} 1 & \text{if } rank_i \leq n, \\ 0 & \text{otherwise.} \end{cases} \quad (41)$$

We name original setting as “Raw” setting. If the newly created triple really exists in the knowledge graph, ranking it before the original triple is not wrong. For example, both of a testing triple (*Pairs*, Located-In, *France*) and a possible corruption (*Lyon*, Located-In, *France*) are valid. In this case, ranking *Lyon* before the correct answer *Pairs* should not be counted as an error. Thus, we take the same filtering operation used in [3] to filter out these triples in training, validation and testing data, which is named as “Filt.” setting. Finally, we remove *Lyon* from the candidate entity list before obtaining the rank of *Pairs* in the above example.

Implementation: As FB15K and WN18 are also used in previous work, the parameters of TransE [3], TransH [27], and TransR [13] are set as the same with the parameters in the published works except that the margin γ of TransH is set to 2 and 3 on FB15K and WN18 respectively. Because this configuration achieves the best results in validation set. For the FB166 dataset, we set the same parameters for TransE, TransH, and TransR with the parameters used in FB15K except that the margin γ of TransH is set to 3. The parameters of our proposed rule-enhanced TransE, TransH, and TransR on three datasets are set as the same with the parameters used in original TransE, TransH and TransR except for the learning rate α on FB15K. The learning rate α of the rule-enhanced TransE, TransH, and TransR on FB15K are set as 0.001, 0.004, and 0.0001 respectively. To avoid overfitting, we initialize the entity and relation embeddings with the results of each enhanced model trained on triples, inference rules and transitivity rules. Based on the initialized entity and relation embeddings, we train the enhanced model on antisymmetry rules by using the new learning rate. Here, the “unif” sampling is used in corrupting negative sample. The optimal parameters are determined by the validation set.

Results: The results on FB166, FB15K, and WN18 datasets are shown in Table 9, Table 10, and Table 11 respectively. KALE was proposed by Guo et al. [8]. The TransE(Pre), TransH(Pre), and TransR(Pre) denote that the triples inferred from rules are used directly as training instances in TransE, TransH, and TransR. The TransE(Rule), TransH(Rule), and TransR(Rule) denote that the rules are jointly used with the original triples. From the results of three datasets, we can see that: (1) The Pre version of TransE, TransH, and TransR models outperform the original TransE, TransH, TransR models which use triples alone on three datasets in most cases. This implies that our rule extraction method can generate correct rules, which can infer new triples. (2) The logic rule-enhanced TransE, TransH, and TransR models greatly improve the performance of the original TransE, TransH, TransR methods which use triples alone. This implies that our rule-enhanced method can efficiently help the translation based knowledge graph embedding models to learn better knowledge representation, e.g., entity embedding and relation embedding. In addition, our rule-enhanced method can encode more information of a rule compared with adding an

Table 8: Examples of rules created.

Dataset	Rule examples
FB	$\forall x, y : \text{location/country/capital}(x, y) \Rightarrow \text{location/location/contains}(x, y)$
	$\forall x, y : \text{location/country/first_level_divisions}(x, y) \Rightarrow \text{location/location/contains}(x, y)$
	$\forall x, y, z : \text{people/person/place_of_birth}(x, y) + \text{location/administrative_division/second_level_division_of}(y, z) \Rightarrow \text{people/person/nationality}(x, z)$
	$\forall x, y : \text{people/family/members}(x, y) \Leftrightarrow \text{people/familymember/family}(y, x)$
	$\forall x, y : \text{location/location/people_born_here}(x, y) \Leftrightarrow \text{people/person/place_of_birth}(y, x)$
WN18	$\forall x, y : \text{_member_of_domain_usage}(x, y) \Leftrightarrow \text{_synset_domain_usage_of}(y, x)$
	$\forall x, y : \text{_hyponym}(x, y) \Leftrightarrow \text{_hypernym}(y, x)$
	$\forall x, y : \text{_part_of}(x, y) \Leftrightarrow \text{_has_part}(y, x)$
	$\forall x, y : \text{_instance_hypernym}(x, y) \Leftrightarrow \text{_instance_hyponym}(y, x)$

Table 9: Link prediction results on dataset FB166.

Method	MR		MRR		Hits@10		Hits@5		Hits@3		Hits@1	
	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
TransE	523	229	0.1679	0.3226	0.3871	0.5707	0.2942	0.4850	0.2083	0.4108	0.0633	0.1825
TransE(Pre)	428	232	0.2426	0.4369	0.4266	0.6586	0.3522	0.5696	0.2769	0.5753	0.1100	0.2938
TransE(Rule)	408	210	0.3413	0.6584	0.5098	0.7392	0.4533	0.7111	0.3996	0.6897	0.2455	0.6100
TransH	431	191	0.2127	0.3332	0.4168	0.5429	0.3358	0.4629	0.2596	0.3970	0.1097	0.2183
TransH(Pre)	415	194	0.1539	0.2033	0.3812	0.4951	0.2845	0.3731	0.2073	0.2879	0.0362	0.0449
TransH(Rule)	407	280	0.3251	0.5818	0.5126	0.7325	0.4523	0.6969	0.3983	0.6640	0.2145	0.4802
TransR	499	185	0.1928	0.4194	0.4344	0.6484	0.3319	0.5711	0.2306	0.5055	0.0839	0.2869
TransR(Pre)	395	218	0.2768	0.4424	0.4457	0.6066	0.3717	0.5299	0.3093	0.4715	0.2033	0.3627
TransR(Rule)	392	152	0.2835	0.6464	0.5011	0.7849	0.4204	0.7486	0.3343	0.7104	0.1642	0.5615
KALE	510	217	0.1897	0.3819	0.4115	0.6061	0.3103	0.5057	0.2247	0.4198	0.0854	0.1947

Table 10: Link prediction results on dataset FB15K.

Method	MR		MRR		Hits@10		Hits@5		Hits@3		Hits@1	
	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
TransE	237	121	0.1575	0.2466	0.3404	0.4646	0.2376	0.3577	0.1711	0.2829	0.0698	0.1376
TransE(Pre)	195	96	0.1872	0.3067	0.3844	0.5332	0.2774	0.4261	0.2073	0.3506	0.0922	0.1931
TransE(Rule)	163	72	0.2923	0.5688	0.5403	0.7779	0.4273	0.7075	0.3404	0.6450	0.1709	0.4517
TransH	227	116	0.1834	0.2827	0.3759	0.5125	0.2706	0.4064	0.2023	0.3322	0.0907	0.1729
TransH(Pre)	235	136	0.1131	0.1572	0.2945	0.3884	0.1916	0.2784	0.1279	0.1985	0.0232	0.0365
TransH(Rule)	175	106	0.2870	0.5170	0.5475	0.7696	0.4328	0.6974	0.3423	0.6207	0.1582	0.3708
TransR	218	72	0.2039	0.3774	0.4385	0.6440	0.3131	0.5332	0.2277	0.4455	0.0967	0.2400
TransR(Pre)	236	153	0.1774	0.2764	0.3165	0.4432	0.2323	0.3521	0.1820	0.2958	0.1084	0.1931
TransR(Rule)	175	55	0.2380	0.4646	0.4724	0.7088	0.3529	0.6119	0.2680	0.5335	0.1284	0.3366
KALE	221	124	0.1864	0.2957	0.3725	0.4956	0.2681	0.3614	0.1841	0.3091	0.0702	0.1491

inferred triple into training data. (3) Our rule-enhanced TransE model outperforms the KALE model [8]. The KALE just encodes rules into TransE model. We implement it using the ParaGraphE. (4) Our rule-enhanced method significantly improves results on the Hits@1 metric, e.g., the Hits@1 increases from 0.086 to 0.7013

on WN18 dataset using rule-enhanced TransE model. The first rank metric is the most important in real knowledge inference task. The Hits@1 score 0.086 indicates that the original TransE model almost cannot rank the the correct triple in the first. Thus, our method can make knowledge graph embedding method capable of being

Table 11: Link prediction results on dataset WN18.

Method	MR		MRR		Hits@10		Hits@5		Hits@3		Hits@1	
	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
TransE	263	251	0.2803	0.3627	0.7256	0.8447	0.5629	0.7148	0.4131	0.5771	0.0564	0.086
TransE(Pre)	236	228	0.3331	0.4234	0.7914	0.8977	0.6610	0.8125	0.5137	0.6882	0.0786	0.1177
TransE(Rule)	231	227	0.7104	0.8189	0.9164	0.9484	0.8893	0.9423	0.8562	0.9351	0.5550	0.7013
TransH	314	298	0.2680	0.3611	0.7546	0.8878	0.5930	0.7764	0.4223	0.6243	0.0250	0.0475
TransH(Pre)	326	316	0.2775	0.3475	0.7913	0.9031	0.6446	0.8023	0.4740	0.6508	0.0014	0.0018
TransH(Rule)	310	310	0.4370	0.5041	0.9025	0.9439	0.8529	0.9290	0.7775	0.8966	0.0861	0.1201
TransR	226	213	0.3307	0.4629	0.7876	0.9295	0.6551	0.8720	0.5034	0.7605	0.0853	0.1428
TransR(Pre)	212	210	0.6512	0.7561	0.8928	0.9415	0.8402	0.9203	0.7599	0.8775	0.5074	0.6244
TransR(Rule)	233	228	0.654	0.7863	0.9068	0.9455	0.8497	0.9286	0.7893	0.9108	0.4984	0.6594
KALE	245	216	0.4203	0.5727	0.7214	0.8659	0.6029	0.7248	0.4731	0.6371	0.0847	0.1463

used in the real knowledge inference task. On the whole, our rule-enhanced method can efficiently leverage the rules to embed the entities and relations into a low-dimensional space.

7.3 Triple Classification

The task of triple classification is to confirm whether a given triple (h, r, t) is correct or not, i.e., binary classification on a triple. It is used in [27] to evaluate TransH model.

Evaluation protocol: We follow the same protocol in NTN [22] and TransH [27]. Evaluation of classification needs negative labels. Thus, we first create labeled data for evaluation. For each triple in the testing or validation set (i.e., a positive triple), we construct 10 negative triples for it by randomly corrupting the entities, 5 at the head position and the other 5 at the tail position⁶. To further make the negative triples as difficult as possible, we corrupt a position using only entities that have appeared in the position. For example, to construct a corrupted triple given a triple (h_o, r, t) by corrupting head entity, the randomly selected head entity h_c must be linked to another tail entity by the relation r . In addition, we need to ensure that each corrupted triple does not exist in either the training, validation, or testing set.

The decision rule for classification is simple: for a triple (h, r, t) , we simply use the dissimilarity score (by the Equation (1), (6), or (18)) to classify triples. If the dissimilarity score is below a relation-specific threshold σ_r , the triple is classified into positive. Otherwise negative. The relation-specific threshold σ_r is set according to the validation set.

Implementation: In the triple classification, the datasets are the same as the datasets used in link prediction task. The search space of hyperparameters is identical to link prediction. Thus, the optimal hyperparameters are set the same as the hyperparameters in the link prediction task.

Results: The accuracies of the triple classification are shown in Table 12. From the results, we can see that: (1) The Per-version based TransE and TransH slightly outperform the original models

⁶Previous works[22, 27] always construct only one negative triple for each testing triple or validation triple. To increase the classification difficulty, we try to use a highly unbalanced setting, with a positive-to-negative ratio of 1:10.

Table 12: Accuracies of triple classification on the testing dataset of FB166, FB15K, and WN18.

Dataset	FB166	FB15K	WN18
TransE	0.9106	0.9094	0.9580
TransE(Per)	0.9097	0.9096	0.9601
TransE(Rule)	0.9490	0.9256	0.9873
TransH	0.9091	0.9086	0.9470
TransH(Per)	0.9093	0.9090	0.9513
TransH(Rule)	0.9505	0.9192	0.9936
TransR	0.9146	0.9091	0.9523
TransR(Per)	0.8727	0.8258	0.9686
TransR(Rule)	0.9204	0.9208	0.9926
KALE	0.9257	0.9011	0.9779

on the three datasets in most cases. (2) Our rule-enhanced method outperforms other models including the original models, the Per-version based model and the KALE model. This once again validates that our method can generate high-quality rules and better leverage the rules in the knowledge graph embedding than baselines.

8 RELATED WORK

Recently, many works have made great efforts on modeling knowledge graphs. Knowledge graph embedding encodes entities of a knowledge graph into a continuous low-dimensional space as vectors, and encodes relations as vectors or matrices. Several works use neural network architectures [2, 22], tensor factorization [6, 17, 20], and Bayesian clustering strategies [24, 31] to explain triples via latent representation of entities and relations. To capture both the first-order and the second-order interactions between two entities, Jenatton et al. [10] proposed Latent Factor Model (LFM) to adopt a bilinear function as its score function. Recently, TransE [3], which models the relation as a translation from head entity

to tail entity, achieves a good trade-off between prediction accuracy and computational efficiency. TransE is efficient when modeling simple relations, such as one-to-one relations. To improve the performance of TransE on complicated relations, various extensions like TransH [27], TransR [13], and TransD [11] are proposed. TransH allows entities to have multiple representations. To obtain multiple representations of an entity, TransH projects an entity vector into relation-specific hyperplanes. TransR also handles the problem of TransE by introducing relation spaces. It allows an entity to have various vector representations by mapping an entity vector into relation-specific spaces. To handle multiple types of relations, TransD constructs relation mapping matrices dynamically by considering entities and a relation simultaneously. However, these models lose the simplicity and efficiency of TransE. To combine the power of tensor product with the efficiency and simplicity of TransE, HOLE [16] uses the circular correlation of vectors to represent pairs of entities.

Most existing methods embed the knowledge graph based solely on triples contained in the knowledge graph. Several recent works try to incorporate other available knowledge, e.g., relation paths [12, 14], entity types [7], Wikipedia anchors [26], and entity descriptions [32]. Logic rules have been widely used in knowledge inference and acquisition [5, 19]. Recent works put growing interest in logic rules. Wang et al. [25] and Wei et al. [28] tried to utilize rules via integer linear programming or Markov logic networks. However, rules are modeled separately from embedding models and cannot help in obtaining better embeddings in these approaches. Rocktäschel et al. [21] proposed a joint model which jointly encodes the rules into the embedding. However, this work focuses on relation extraction task and learns the entity pair embedding instead of entity embeddings. In [8] a new method named KALE is proposed to jointly embed the knowledge graph and logic rules. The rules are represented as complex formulae and are modeled by the t-norm fuzzy logics.

9 CONCLUSION AND FUTURE WORK

In conclusion, we present a novel logic rule-enhanced method which can be easily integrated with any translation based knowledge graph embedding model. Besides that, an automatic logic rule extraction method is introduced in this work. Our method places the logic rules and triples in the same space. Our experiments show that our proposed method can efficiently improve the performance of knowledge graph embedding in various tasks. Particularly, it brings a great improvement in the filtered Hits@1 evaluation which is a pivotal evaluation in the knowledge inference task. This makes the knowledge graph embedding method capable being applied in the real and large scale knowledge inference tasks.

This is our first step in the scope of external knowledge powered knowledge graph embedding. In the future, we will explore more types of effective rules that exist in data. Besides, we plan to handle the embedding of multiple knowledge graphs, such as jointly embedding Freebase and YAGO.

REFERENCES

- [1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *In Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), 1247–1250.
- [2] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A Semantic Matching Energy Function for Learning with Multi-relational Data. *Machine Learning* (2014), 233–259.
- [3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Ok-sana Yakhnenko. 2013. Translating embeddings for modeling multirelational data. *In Advances in Neural Information Processing Systems* (2013), 2787–2795.
- [4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. *In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011), 301–306.
- [5] Matthias Bröcheler, Lilyana Mihalkova, and Lise Getoor. 2014. Probabilistic similarity logic. *In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence* (2014), 73–82.
- [6] Kai-Wei Chang, Wen-Tau Yih, Bishan Yang, and Chris Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014).
- [7] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), 84–94.
- [8] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly Embedding Knowledge Graphs and Logical Rules. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016), 192–202.
- [9] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*. (2016).
- [10] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. 2012. A latent factor model for highly multi-relational data. *In Proceedings of the 25th International Conference on Neural Information Processing Systems* (2012).
- [11] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), 687–696.
- [12] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), 705–714.
- [13] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015), 2181–2187.
- [14] Yuanfei Luo, Quan Wang, Bin Wang, and Li Guo. 2015. Context-dependent knowledge graph embedding. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), 1656–1661.
- [15] George A. Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.
- [16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2015. Holographic embeddings of knowledge graphs. *Computer Science* (2015).
- [17] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. *In International Conference on Machine Learning* (2011), 809–816.
- [18] Xiao-Fan Nie and Wu-Jun Li. 2017. ParaGraphE: A Library for Parallel Knowledge Graph Embedding. *arXiv* (2017).
- [19] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. *In Proceedings of the 12th International Semantic Web Conference* (2013), 542–557.
- [20] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. *In Proceedings of NAACL-HLT* (2013), 74–84.
- [21] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. *In Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2015).
- [22] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *In Advances in Neural Information Processing Systems* (2013), 926–934.
- [23] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. *In Proceedings of the 16th international conference on World Wide Web* (2007), 697–706.
- [24] Ilya Sutskever, Ruslan Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2009. Modelling relational data using bayesian clustered tensor factorization. *In Proceedings of the 23rd Annual Conference on Neural Information Processing Systems* (2009), 1821–1828.
- [25] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. *In Proceedings of the 24th International Conference on Artificial Intelligence* (2015), 1859–1865.

- [26] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014), 1591–1601.
- [27] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014), 1112–1119.
- [28] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengyu Sun, and Guanhua Tian. 2015. Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), 1331–1340.
- [29] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), 1366–1371.
- [30] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (2012), 481–492.
- [31] Zhao Xu, Volker Tresp, Kai Yu, and Hans Peter Kriegel. 2006. Infinite hidden relational models. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence* (2006), 544–551.
- [32] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), 267–272.